MASTER THESIS

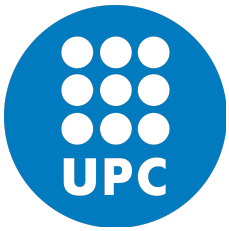
Automatic load-balance method for coupled Earth System Models

Author:
Sergi Palomas

Directors:
Dr. Mario C. Acosta
Dr. Etienne Tourigny

Supervisor:
Dr. Carlos Álvarez

Thesis presented for the degree of
Master in Innovation and Research in Informatics
High Performance Computing



Universitat Politècnica de Catalunya
June, 2022

Abstract

Earth System Models (ESMs) are complex models used to simulate the Earth climate and are commonly built from different independent components that simulate a specific natural phenomenon (ocean dynamics, atmospheric dynamics, atmospheric chemistry, land and ocean biosphere, etc.). To simulate the interactions between these processes, ESMs use coupling libraries that manage the synchronization and field exchanges between the independent components, running in parallel in a typical Multi Program, Multiple Data (MPMD) application. The performance achieved depends on the coupling approach, and on the number of parallel resources and scalability properties of each component. Finding the best number of resources to use for each component of coupled ESMs is crucial to use the parallel resources efficiently. However, it is still a task involving manually testing multiple process allocations by trial and error, leading to configurations that are sub-optimal given that the dependencies between the constituents are complex and models do not scale perfectly. This project presents a methodology to find the optimal number of resources to allocate for each component to achieve the best computational performance for the coupled ESM, minimizing the cost of executing each of the constituents, which may not run at individual optimal configurations, and the waiting time due to the synchronizations between them. To achieve this, a number of novel metrics were designed and implemented in order to identify the component(s) acting as a bottleneck(s) and to evaluate the performance of the coupled execution according to different Energy-To-Solution (ETS) / Time-To-Solution (TTS) tradeoff criteria. The methodology has been tested against multiple resource configurations used for the widely known ESM in Europe: EC-Earth3. The results show that some configurations could run up to 34% faster and reduce the execution cost by 6.7%. Moreover, the method has been contrasted against a configuration used for the Coupled Model Intercomparison Project Phase 6 (CMIP6)) and achieved a set-up 5% faster and 1% less costly. Lastly, the work has been integrated into a workflow manager to automatize the tasks, involving minimum user intervention.

Contents

Acronyms

AS Autosubmit workflow manager

BSC Barcelona Supercomputing Centre

CGCMs Coupled General Circulation Models

CHSY Core-Hours per Simulated Year

CMIP6 Coupled Model Intercomparison Project Phase 6

CPMIP Computational Performance Model Intercomparison Project

ECMWF European Centre for Medium-Range Weather Forecasts

EDP Energy-Delay Product

ESMs Earth System Models

ETS Energy-To-Solution

ETS$_r$ Energy-To-Solution ratio

EUCP European Climate Prediction project

FN Fitness metric

HR High Resolution

IFS Integrated Forecasting System

MPMD Multi Program, Multiple Data

NEMO Nucleus for European Modelling of the Ocean

PEs Processing Elements

RNF Runoff-mapper

SR Standard Resolution

SYPD Simulated Years Per Day of execution

TTS Time-To-Solution

TTS$_r$ Time-To-Solution ratio

XIOS XML Input Output Server

Chapter 1

Introduction

Coupled General Circulation Models (CGCMs) have been used to simulate the Earth system since the late 1960s (Manabe S. et.al [1]) including numerical codes simulating the atmosphere, the ocean, the land surface and the sea ice. The standard constituents of a CGCM model are an Atmospheric General Circulation Model (AGCM) including a land surface model coupled to an Oceanic General Circulation Model (OGCM) incorporating a sea-ice model. Developing climate CGCMs that are capable of simulating the Earth system over a wide range of spatial and temporal scales is one of the main objectives of the World Climate Research Programme (WCRP) of the World Meteorological Organisation (WMO), aiming to "facilitate analysis and prediction of Earth system variability and change for use in an increasing range of practical applications of direct relevance, benefit and value to society"[1].

To facilitate and improve the quality of the analysis and prediction, the scientific community organized the Coupled Model Intercomparison Project (CMIP), started 20 years ago as a comparison of coupled climate models. These models are used to run experiments using atmosphere components coupled to a dynamic ocean, a simple land surface, and thermodynamic sea-ice (Eyring V. et al. [2]). Throughout the years, it has evolved over 6 phases into a major international multi-model research activity that has become crucial for the climate science research and for the national and international assessments of climate change. In the latest phase, the Coupled Model Intercomparison Project Phase 6 (CMIP6) comprised 23 individually designed Earth System Models (ESMs) (Gerald A. et al. [3]). The total amount of output from CMIP6 was estimated to be between 20 and 40 petabytes of experiments with various resolutions and configurations. Accounting only for the production runs, almost 240000 years were simulated, as shown by a collection of the computational performance metrics of various institutions and models by Acosta M. et al. [4].

The scientific community acknowledges that climate models need to evolve to be more complete, accurate and complex. As simulating more features of the Earth system has shown to improve the usefulness of climate models, more components have gradually been included to simulate more natural climate phenomena (e.g., atmospheric chemistry, marine biology, the carbon cycle). However, the introduction of more components also implies more complexity. Given the nature of the underlying physics, components within the system have to interact via coupling and the more components, the more complex will be the coupling.

These kinds of simulations using different components are known as coupled ESMs. There are different approaches when it comes to the design of the coupling process.

[1] http://www.wcrp-climate.org/mission.shtml

Often, multiple individually developed codes are executed concurrently and synchronized during the runtime to exchange fields with the others. This kind of applications are computationally known as Multi Program, Multiple Data (MPMD) and components executed in parallel can use different parallel paradigms such as MPI to take advantage of the computational resources of HPC machines.

Achieving a good performance of coupled ESMs is a difficult task. MPMD applications are made of multiple independent codes, each one having its own performance issues and parallelism characteristics. Additionally, the grids used can be very different between the components and extra computation and even serialization is needed to interpolate (regrid) the data, reducing the overall parallel efficiency. One of the most problematic issues that coupled ESMs introduce is the load imbalance. Coordination among components is required to exchange coupling fields, which means that faster components have to wait for the slower ones before continuing their own execution. All the computing resources given to a component that has to wait will be idle. Minimizing this waiting time (or rather the cost of this idle processes) is known as load-balancing the coupled ESM. Finding the best resource configuration is a non-trivial process that involves: analyse the individual components speedup at different processor counts, study the interaction between them during the coupling and make compromises between the cost and time to execute the coupled ESM.

Chapter 2

Related work

ESMs are among the most challenging applications that run on HPC platforms. The difficulties that they present now and for the upcoming generation of models (with even higher resolutions and data intensity) and machines (exascale computing) are addressed by André J. et al. 2014 [5], Cappello F. et al. [6], Attig N. et al. [7], Reed D. and Dongarra J. [8], and Wehner et al. [9]. On the current generation of new machines, ESMs have been able to show only modest performance gains as shown by Balaji V. et al. in [10]. One of the main factors that limit ESMs performance is the load-imbalance. Valcke S. et al [11] have shown that most of the climate and weather applications used in a wide range of institutions are set to run multiple individual components in parallel, and the interactions between them are handled by a coupler. There are two different approaches to couple different components: using an "external coupler or coupling library", and the "integrated coupling framework" [12]. In the first approach, the component models remain as separate executables and the original codes are modified as little as possible. The code of the component models is instrumented with calls to the coupling library API and the synchronization of the components is implicitly ensured by the algorithm of the coupling exchanges. The communication, regridding and other transformations on the coupling fields are done either directly by the coupling library or by a subset of processes especially allocated for the coupler to do the work. OASIS [13], MCT [14], C-Coupler [15], and YAC [16] are some examples. The integrated coupling framework approach involves splitting the original component codes into initialize, run and finalize units. Adapting them to standard data structures, routine interfaces and rebuilding a single integrated application based on these units. The coupler layer is in charge to call the different component and coupling units and controls their execution. Some examples are the Factorial snowpack model (FSM) [17], the Earth System Modeling Framework (ESMF) [18] and Cpl7 [19].

One model using integrated coupling framework approach is EC-Earth3 [20], a global ESM used in multiple institutions in Europe which uses the OASIS3-MCT coupling library to control the information sent and received by the constituents. In this context, different components are synchronized during their execution and faster components have to wait for the slower ones. Trying to find a combination of processors that minimizes the cost of having dependencies between components is known as load-balancing an ESM. Load-balance algorithms can be divided in dynamic load balancing, where the load-imbalance is minimized during the runtime, and static load-balancing, where the process involves stopping and rerunning the model execution to find resource configurations that minimize the coupling cost. Most of the literature focuses on solving the load-imbalance dynamically. But it is not a feasible solution for big ESMs which are constituted by individual models written and maintained by different institutions in the

short term, given that the source code of this type of ESMs can not be freely modified and the coordination to apply malleability to all of the constituents would take too much effort. This is why in these circumstances, static solutions are preferred.

To deal with the load-balance problem "dynamically" some scientists have explored the possibility of reallocating the processes on which an application runs during its execution. A property known as malleability [21]. Vadhiyar and Dongarra developed SRS [22], a framework that allows a parallel application to reconfigure the number of processes that it uses by stopping and restarting its execution using checkpoints. Maghraoui et al. [23, 24] implemented malleability integrating the Internet operating system (IOS), a framework for middleware-driven dynamic application reconfiguration, to the PCM (process checkpointing and migration) library. This approach proves the usefulness of malleability to deal with the load-balance problem using the PCM/IOS framework. But while they allow to reconfigure the processes on which an MPI application runs, the overhead of checkpointing and resuming the execution is too high. Another example of dealing with the load-balance issue "dynamically" by taking profit of the malleability of the constituents is the load balance manager (LBM) module, which has been developed in the Malleable Model Coupling Toolkit (MMCT) by Kim D. et al. [25]. MMCT is an extension of MCT (Model Coupling Toolkit, Larson J. et al. [14]) that supports dynamic load balance in parallel coupled models. It consists of MCT, LBM and a dynamic process and communicator management system. The LBM decomposes the time of each component during a coupled interval (CI, interval after which the coupled system has evolved) as *constituent computation* and *constituent coupling*. The LBM will tag components as *donor* (the fastest constituent) or *recipient* (the slowest constituent) and will keep reallocating the Processing Elements (PEs) away from the first to the second at each coupling interval until the solution stops improving. However, the solution converges too slowly to be used in production. Kim D. et al. [26] proposed an improvement for LBM consisting of a Predictive load-balancing algorithm. After each CI, the constituent time and number of resources used are stored into a file. After some iterations, the LBM is able to predict the constituent time using N processors by reading its constituent record. Finally, it uses that value for all constituents to predict the CI execution time under a particular resource configuration. This method was extended by Kim D. et al. [27] to support coupled models that have changing loads during the execution. Lastly, Kim et al. [28] enhanced the performance predictor by instrumenting each model and obtaining the timing and interaction information automatically. Then, the predictor can relate CI execution time with the execution time of each individual model and guide the PEs reallocation automatically. However, the results are very sensitive to the timings that are captured during the runtime, which are taken for a very short execution period (a single CI). It is well known that applications running in HPC machines will have some degree of variability and it is better to record the performance of an application running in tens or hundreds of nodes during long simulations and, preferably, replicating the measures. Furthermore, the proposed method is based on the ability of changing the number of processes that a constituent uses during the runtime, a feature accessible only if the models are ported to MMCT. Furthermore, it has only been tested in toy models and not in big ESMs such as EC-Earth3.

Another approach for the load-balance problem was proposed by Alexeev Y. et al. [29], Nan D. et al. [30] and Hongliang et al. [31] to avoid manually setting the number of processes for each component of the Community Earth System Model (CESM) by formulating the problem as a mixed-integer nonlinear optimization problem. The component, layout and process count (integer) are the decision variables, and the component running time is a nonlinear function of the process count. Their scheduling is based in

the fact that CESM coupling approach allows components to be executed sequentially in the same processors, which is done as subroutines instead of independent binaries with a static number of resources during the whole run. Furthermore, the predictions are based on performance models and not from instrumenting and running "real" simulations.

Although the examples presented previously are considered as dynamical solutions according to the state of the art, it is important to highlight that they are static solutions at the end. In these previous examples, the new number of resources is changed after localizing the optimal by getting the performance metrics during the runtime. However, a real dynamical solution should be able to change and load balance the work along all the iterations of a complete simulation, using other kind of approaches as tasks, an option that has been explored in the Dynamic Load Balance[1] library developed at the Barcelona Supercomputing Centre (BSC). This library has been used to speedup hybrid parallel applications and maximise the utilization of computational resources. It improves the load-balance of hybrid applications managing the number of threads in the shared memory level. One of its modules, Lend When Idle (LeWI) has been used to reassign computation resources of blocked processes to other processes more loaded successfully on MPI+OpenMP and MPI+SMPSs hybrid applications by Garcia M. et al. in [32, 33]. This could not only be used to deal with the coupled load-balance problem, but also to solve imbalances inside the tasks within a component. Although this is a promising option to be explored in the future, it can not be considered as the only solution. It is not possible to ensure that an ESM will use a parallelization paradigm suitable for tasks (such as OpenMP). Additionally, the work done until now using the Dynamic Load Balance library with hybrid MPI+OpenMPI parallelization proves that the library has room for improvement and it is not ready for operational applications as the ESM under study in this work.

On the other hand, there are the static solutions. An example of dealing with the load-balance statically for an ESM was shown by Will A. et al. in [34] for the COSMO-CLM regional climate model. This model uses OASIS3-MCT to couple the land surface models, Mediterranean, North and Baltic seas models, and the MPI-ESM Earth system model. The LUCIA tool was used to find the optimum processor count for each component considering the Time-To-Solution (TTS), computing cost and parallel efficiency of the simulation. They stated that given a number or PEs, the coupled TTS is minimized if all components have the same execution time. Which ensure that there is no load imbalance (no idle cores during the simulation). The method consisted on, once a proportion of PEs per component that minimized the coupling cost was found, double the resources for each component (and doing some re-distributing if necessary) until one of the components reaches a limit of 50% of parallel efficiency. Although this seems to be a good solution at first (actually it is the most adopted one by the earth system community), in this work we will show that running all the constituents at the same speed can lead to sub-optimal solutions. Furthermore, there is no value that can be used as a threshold for the parallel efficiency that ensures optimality given the heterogeneity of models and platforms.

For EC-Earth3, the ESM under analysis in this work, Acosta M. et al. showed in [35] the performance issues due to the coupling. Identified some caveats with the LUCIA metrics that difficult the task of load-balancing the atmospheric and oceanic coupled configurations and demonstrated that the coupling time can be very high when conservation methods are used in the coupler.

Performance metrics are crucial when evaluating how models of such complexity run

[1] https://pm.bsc.es/dlb

on HPC machines. Traditional metrics, like the floating-point operations per second, are not fully representative for multi-physics models like the ones related to climate and weather forecasting. Balaji V. et al. [36] also showed that a single performance metric is insufficient given the heterogeneity of ESMs, proposing the Computational Performance Model Intercomparison Project (CPMIP) metrics, which are a collection of metrics universally available, representative of the performance under "real" conditions (not under ideal conditions nor from subsets of the code), which can cover data and computational load and are easy to collect. One of the metrics to evaluate the load-imbalance is the coupling cost. We have reformulated this metric to, instead of giving the overall execution cost loss due to coupling events, provide how much each component adds to the coupling cost (the Partial coupling cost). This metric will help to identify which is the component acting as a bottleneck in coupled ESMs. The Energy-Delay Product (EDP) was a metric proposed by Gonzalez R. et al. [37] to evaluate the time/energy trade-off on general purpose microprocessors. In the HPC field, the EDP metric is still used although the naming can be different: Abdulsalam S. et al. called it Powerup in [38] and Yepes-Arbós X. et al. [39] named it as "Performance-efficiency compromise". In this work we introduce the Fitness metric (FN): a parametrizable performance metric that allows to find different optimal solutions (number of processes to allocate) depending on the selected Energy-To-Solution (ETS) / TTS tradeoff criteria.

Chapter 3

Problem to solve, objectives and approach proposed

The load-imbalance is known to be one of the main factors limiting the performance of coupled ESMs. While some works have proposed to deal with the problem by reallocating the resources during the run (dynamic load-balance), they are not suitable for many ESMs since malleability is currently not an option. Therefore, the optimal resource configuration has to be found by manual approaches using static load-balancing. Without a methodology to minimise the load-imbalance and a set of metrics designed to evaluate and understand the performance of coupled ESMs, the best possible solution can not be found. One of the most widespread approaches is to find a resource configuration in which all components run at the same speed, expecting that this method will reduce the waiting time to synchronize the models to the minimum. This, however, can lead to suboptimal solutions under some circumstances. We will show in our analysis that there are better resource configurations which may seem counter-intuitive at first, but are easy to identify with the right performance metrics (more in Section 5.2). The objectives of this work can be summarized as follows:

- Create a methodology to find the the best possible resource configuration for coupled ESMs, which doesn't involve modifying the sources of the models but only changing how many PEs are allocated to each one.
- Define a metric to evaluate the performance of coupled ESMs that allows to have control over the energy cost / execution time tradeoff.
- Integrate the steps into a workflow manager so that finding the optimal result requires minimum user intervention.

The novel developments that this work has introduced are:

- The Scalability workflow (Section 6.1): A new workflow that automatizes the process of getting the scalability curves of individual constituents. A single coupled run will have the information of each of the constituents execution time as if they were executed standalone.
- The Prediction script (Section 6.2): A Python script that, given the scalability curves of the constituents, will predict the best combination of PEs subject to user constraints like the energy/time tradeoff criteria, restriction on the number of PEs to use and maximum parallelization allowed for the coupled run.

- The Load-balance workflow (Section 6.3): A new workflow that runs multiple resource configurations for the ESM on the HPC machine, gets the performance metrics and tries to improve the solution even further by changing the PEs allocation based on the results from real simulations.

The new methodology will be used to optimize resource configuration for different resolutions of EC-Earth3 simulations. The results will be compared to the default allocations used on the European Centre for Medium-Range Weather Forecasts (ECMWF) machine, and to the ones used by large projects such as CMIP6 and the European Climate Prediction project (EUCP) executed in MareNostrum4. The method proposed has shown to provide resource configurations that are up to 37% faster and reduce the execution cost by 6.7%.

Chapter 4

Context

Climate science is, by nature, a complex problem where we try to simulate the time evolution of the Earth system. ESMs have always presented issues regarding their performance on HPC platforms. This problem comes from the inherent complexity of the models, and because the development involves many independent teams that are specialized in one of the sub-parts (e.g. ocean dynamics, atmospheric chemistry, biosphere, etc.). ESMs often are built of a mixture of independent codes that make multi-component ESM run separate binaries concurrently with their connections managed by a coupler, raising the load-balance problem. This problem comes when coupled components have to be synchronised during the simulation and if they do not run at the same speed, some processors will be idle (waiting for the slowest component).

For a coupled model, components may not be able to run at their individual optimal scaling point but at the point which is optimal for the ESM as a whole, which also takes into account the performance loss due to synchronizations. Finding the optimal number of processes for one component involves testing the throughput of that model at different PEs counts and use a metric like the EDP to numerically evaluate each one of the scalability points. Reducing the load-imbalance involves measuring the cost of the dependencies between components and modify the number of PEs allocated to each one accordingly.

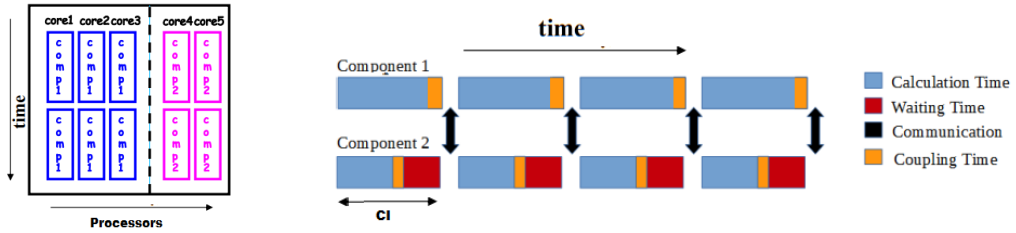
The next sections give an overview of the performance issues of the different coupling approaches, introduce the ESM under study (EC-Earth3) and its most used configurations (atmosphere component coupled with the ocean), and provide a brief description of the workflow manager used (Autosubmit).

4.1 Coupling approaches

As mentioned, there are two ways in which different components can be coupled: using an "external coupler or coupling library" where components remain as separate executables and use a coupling library, and the "integrated coupling framework" where the original component codes are modified and redefined to be part of a single larger application. The external coupler or coupling library approach will generally reduce the efficiency of the ESM. Components involved in the simulation have to run concurrently on separate processors. The coupling exchanges occur between different processes/nodes and the speed at which they can be transferred depends on the bandwidth and latency of the HPC network. Also, as components have to be synchronized to exchange the fields, data dependencies that were not present in stand-alone runs appear. Processes of a component waiting for a field from another component will be idle and reduce the ESM

parallel efficiency. Figure 4.1 illustrates two components coupled using this approach. Figure 4.1a shows that each component runs as an independent binary and has its own PEs allocated during the whole simulation. Figure 4.1b shows the pattern of this approach during the simulation. Both components are synchronized at every coupling interval (CI). Component 2 is faster than Component 1 (smaller Calculation time) and has to wait at the end of each iteration. The execution time of both components is extended at the end of their steps when they have to interpolate the fields before sending them to a component using a different grid or due to the execution of conservative algorithms to ensure that the total integrated value of a coupling field is the same in both the source and target grid.

On the contrary, in the integrated coupling framework the same processor can run the multiple components as routines. This offers the possibility to make components share the coupling fields using the DRAM. Furthermore, components can be allowed to be flexibly placed on arbitrary subsets of computing resources in a sequential, concurrent or mixed mode. Helping to define an overall sequencing of the components that optimizes the coupled ESM performance. Figure 4.2[1] shows some of the possible layouts that are supported for some couplers using this approach.



(a) (b)

Figure 4.1: Overview of two components using a coupling library.

[1] https://is.enes.org/events/is-enes3-webinar-on-oasis3_mct

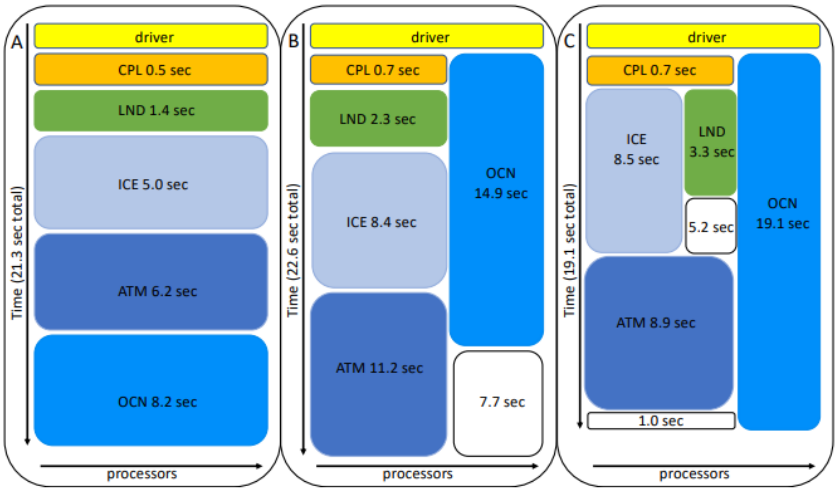


Figure 4.2: Layouts supported in Cpl7 using the integrated coupling framework approach. In A (left), all components (Coupler CPL, land LND, ice ICE, Atmosphere ATM and ocean OCN) are run sequentially on all available processors. In B (center), the OCN is run in parallel with the other components, which themselves run sequentially on a subset of the remaining processors. Some idle time after the OCN execution. This is so called the hybrid layout. In C (left), the hybrid layout is used again but achieving a better load-balance by changing the placement of components to the computing resources.

4.2 EC-Earth3

EC-Earth3 is a global coupled climate model, which integrates a number of component models in order to simulate the Earth system. It is developed by a consortium of European research institutions, which collaborate in the development of a new ESM. The goal of EC-Earth3 is to build a fully coupled atmosphere-ocean-land-biosphere model usable for problems encompassing from seasonal-to-decadal climate prediction to climate change projections and paleoclimate simulations. Figure 4.3 shows an overview of a configuration involving 5 components and an IO server and how they are interconnected by the OASIS3-MCT coupler. A brief description of the components is listed below:

- The OASIS3-MCT coupler: a coupling library to be linked to the component models and whose main function is to interpolate and exchange the coupling fields between them to form a coupled system.

- The Integrated Forecasting System (IFS) as atmosphere model: an operational global meteorological forecasting model developed and maintained by the European Centre of Medium-Range Weather Forecasts (ECMWF). The dynamical core of IFS is hydrostatic, two-time-level, semi-implicit, semi-Lagrangian and applies spectral transforms between grid-point space and spectral space. In the vertical the model is discretized using a finite-element scheme. A reduced Gaussian grid is used in the horizontal.

- The Nucleus for European Modelling of the Ocean (NEMO) as ocean model: a state-of-the-art modelling framework for oceanographic research, operational oceanography seasonal forecast and climate studies. It discretizes the 3D Navier-Stokes equations, being a finite difference, hydrostatic, primitive equation model,

with a free sea surface and a non-linear equation of state in the Jackett. The ocean general circulation model (OGCM) is OPA (Océan Parallélisé), a primitive equation model which is numerically solved in a global ocean curvilinear grid known as ORCA. EC-Earth 3.3.2 uses NEMO's version 3.6 with XML Input Output Server (XIOS) version 2.0, an asynchronous input/output server used to minimize previous I/O problems.

- The Louvain-la-Neuve sea-Ice Model 3 (LIM3): a thermodynamic-dynamic sea-ice model directly coupled with OPA.
- PISCES-v2 (Pelagic Interactions Scheme for Carbon and Ecosystem Studies volume 2) : an ocean biogeochemical model for nutrients and the carbon cycle that is directly coupled to NEMO.
- LPJ-GUESS (LPJG) as a dynamic vegetation-terrestrial ecosystem model.
- The Tracer Model 5 (TM5): a global chemistry transport model describing the atmospheric chemistry and transport of reactive or inert tracers.
- The Runoff-mapper (RNF) component: used to distribute the runoff from land to the ocean through rivers. It runs using its own binary and coupled through OASIS3-MCT.

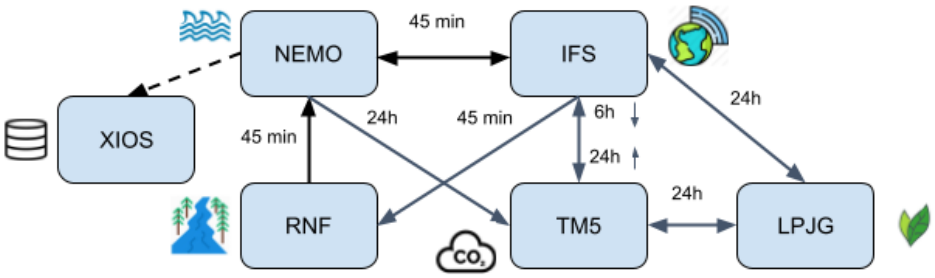


Figure 4.3: Overview of an EC-Earth3 experiment using NEMO as the ocean (with sea ice and ocean biogeochemistry), IFS as the atmosphere, LPJG as the vegetation, TM5 as the chemistry, RNF as the runoff from land to the ocean and XIOS as the IO server. The arrows show the dependencies between components in simulated time. XIOS does not communicate through OASIS-MCT but adds some dedicated processes (IO servers) directly connected to NEMO to handle the ocean output in parallel.

The configurations under study for this work will be the Standard Resolution (SR) and High Resolution (HR) simulations. They are the most used on EC-Earth3 and, therefore, the ones that consume more computing resources and for which any gain in performance has a greater impact. They both include IFS coupled with NEMO as the main components, parallelized using MPI, and which interchange 23 fields (6 from NEMO to IFS and 17 from IFS to NEMO) through OASIS3-MCT at the beginning of their own timestep. As a consequence, the two components have to be synchronized before starting executing their own computation.

In the SR, IFS uses the T255L91[2] grid, which corresponds to a resolution of 80 km for the atmosphere, coupled to NEMO using an ORCA1L75[3] grid, which corresponds

[2] https://confluence.ecmwf.int/display/OIFS/4.+OpenIFS%3A+Grid+and+Resolution
[3] https://www.nemo-ocean.eu/doc/node108.html

to a 1 degree resolution at the equator (∼25km). In the HR configurations, the grids are T511L91 for the atmosphere and ORCA025 for the ocean, which correspond to a resolution of 40 km and 1/4 of a degree for IFS and NEMO, respectively. They both involve, in addition to NEMO and IFS the RNF and XIOS components. For the load-balancing, XIOS and RNF are not taken into account, as XIOS does not communicate via OASIS but directly with NEMO to handle its IO operations in parallel and RNF runs in serial and is much faster than the other components.

4.3 Autosubmit

Workflow managers are used to control the tasks and their dependencies in complex Earth system workflows, which normally include running more tasks than just the simulation: the models need to be compiled on the HPC machine, the data for the initial conditions and namelists to configure the run have to be present in the runtime directory, the output from the simulation is often post-processed and the data is normally moved out of the HPC filesystem once the experiment has finished. Scientists at BSC and other institutions use the Autosubmit workflow manager (AS) [40] to do these tasks with EC-Earth3.

Autosubmit [40] (AS) is a Python-based workflow manager that helps creating, running and monitoring climate experiments remotely using computing resources on supercomputing platforms. It is able to handle complex tasks involving different jobs. These jobs can be executed locally or remotely on different platforms (e.g. HPC computing nodes, data transfer nodes, workstations). It manages the submission of jobs to the queue scheduler, provides the tools to control job status and to build their dependencies and can serve to handle errors during the workflow execution. It was specifically developed with climate experiment's needs in mind and it is currently used at BSC to run several climate, weather and air quality models such as EC-Earth3, MONARCH, NEMO, CALIOPE and HERMES on numerous HPC platforms for research and production.

As in this work the model of interest is EC-Earth3 and it has a production workflow running with AS, integrating the load-balance method to this workflow manager will also serve to make it accessible to other users and portable to other machines without the extra effort that it would require otherwise.

Chapter 5

Coupled ESM performance

5.1 Performance metrics to evaluate coupled ESMs

Measuring the performance of MPI applications (like EC-Earth3) on HPC machines involves timing the execution of the application running at different processor counts (i.e. getting the strong scalability curve). This helps to identify how well the program scales by comparing the results against the ideal speedup and find the maximum speed at which the application can run. As a general rule, one would like to see that when doubling the number of processors of the run, the TTS drops by a half. However, it is well known that models scale less than perfectly. Therefore, boldly selecting the number of resources that maximises the speed is usually a bad decision as it leads to a waste of resources. Energy metrics are used to evaluate the cost associated to the execution and they increase when adding more resources under a non-perfect scalable model. Selecting an appropriate number of resources to execute the program becomes, therefore, a trade-off between the speed and the execution cost.

5.1.1 CPMIP performance metrics

The set of performance metrics for the computational performance model intercomparison project (CPMIP) try to take into account the structure of ESMs and how they are executed in production runs. The ones of interest for this work are listed below:

- **Parallelization** (NP): Total number of cores allocated for the run
- **Simulated Years Per Day of execution (SYPD)**: The simulated years (SY) per day of execution (24h of execution time on the HPC) of the ESM
- **Core-Hours per Simulated Year (CHSY)**: The core hours per simulated year. Measured as the product of the model runtime for 1 SY and the number of cores allocated (NP). Note that the CHSY and SYPD are related by the following formula:

$$CHSY = \frac{24 \cdot NP}{SYPD} \tag{5.1}$$

- **JPSY**: The energy cost of a simulation, measured in Joules per simulated year. Directly proportional to the CHSY.
- **Coupling cost** (Cpl_cost): Measures the overhead caused by the coupling. This can be due to the waiting time caused by the synchronization between models within the ESM (faster components have to wait for slower ones), the added cost

of interpolating the data from the source grid to the target one and the time spent in communications when sending/receiving the data.

$$Cpl\_cost = \frac{T_M NP - \sum_c T_C P_C}{T_M P_M} \tag{5.2}$$

Where T_M and NP are the runtime and parallelization for the whole model, and T_C and P_C the same for each component.

For this work, the Equation 5.2 has been reformulated to evaluate how much each component adds to the coupling cost, which is essential to know which component should lend PEs, and which one should receive them. It has been called Partial coupling cost:

$$Partial\_cpl\_cost = \frac{T_{Ccpl} P_C}{T_M \cdot NP} \tag{5.3}$$

Where T_{Ccpl} is the total time spent by a component in coupled events (waiting, interpolating and sending).

During the simulation, the OASIS3-MCT coupling library will record the timings for the starting and ending of waiting (Oasis_get), sending (Oasis_put) and interpolation (Oasis_interpo) events for each component involved. After the run, this timing information is post-processed using the LUCIA[1] tool to collect the mentioned CPMIP metrics for the simulation.

5.1.2 Time-to-Solution vs Energy-to-Solution criteria

If we want an application to run faster, we will increase the number of PEs and, consequently, the cost (i.e. energy) of the execution will be higher. Given that some of the ESM components can not run in serial, the execution in a single node (P_o) is taken as the baseline. Therefore, the Speedup will be defined as:

$$Speedup = \frac{T_{P_o}}{T_P} \tag{5.4}$$

Where T_P is the execution time using P processes. Consequently, the parallel efficiency will be:

$$Efficiency = \frac{Speedup}{\frac{P}{P_o}} \tag{5.5}$$

Probably one of the most widespread metrics to evaluate the performance of a program is the Energy-Delay Product (EDP). Which for MPI applications can be computed as:

$$EDP = \frac{Speedup}{Efficiency} \tag{5.6}$$

For this work, we define a new metric that allows to parameterise the ratio of Time-Energy at which we want to run: the Fitness metrics FN. We first define a Time-To-Solution ratio (TTS_r) and the Energy-To-Solution ratio (ETS_r), both of which range between 0 and 1 and must add to unity:

$$TTS_r + ETS_r = 1 \tag{5.7}$$

[1] https://www.cerfacs.fr/globc/publication/technicalreport/2014/lucia_documentation.pdf

Then, given the scalability curve of one component, where for each core count we have the SYPD (metric of speed) and CHSY (metric of cost), the FN is calculated as

$$FN = TTS_r \cdot SYPD_n + ETS_r \cdot (1 - CHSY_n) \tag{5.8}$$

Where $SYPD_n$ ($CHSY_n$) is the value of the SYPD (CHSY) after a min-max normalization. Note that we use $1 - CHSY_n$ given that the greater the cost, the less energy efficient the execution will be (i.e. the minimum cost maximizes the energy).

By pondering the SYPD (CHSY) with the TTS_r (ETS_r) ratio values, the result is more flexible and allows finding an optimal resource configuration depending on the needs.

Table 5.1 shows how the FN metric compares to the EDP under varying TTS_r for IFS in SR. The $SYPD_n$ ($CHSY_n$) column is the value of the SYPD (CHSY) after a min_max normalization. For instance, using 48 PEs for IFS is the slowest configuration ($SYPD_n$=0) but the one that consumes less energy ($1 - CHSY_n$=1). On the other hand, using 1008 PEs is the fastest configuration ($SYPD_n$=1) but the worst in terms of energy ($1 - CHSY_n$=0).

Using the EDP metric, the optimum is at 864 PEs (576 PEs being a very close second). Using the FN metric, we see that as we increase the TTS_r, the optimum tends to be a resource configuration with a larger number of PEs. For a $TTS_r \leq 0.3$, the solution is to use the minimum number of resources possible, as it's the configuration which minimizes the CHSY. For a TTS_r close to 0.5, the metrics selects configurations that are balanced in terms of speed and energy cost. For a TTS_r of 0.7 the best configuration matches the one we obtained with the EDP metric. Finally, if the TTS_r is set over 0.9, the best configuration is the one that maximizes the SYPD, no matter the cost. Note that the FN metric can asses that not all PEs counts are optimal. For instance, changing the TTS_r from 0.6 to 0.7 finds two optimal solutions that are $864 - 576 = 288$ PEs apart.

Figure 5.1 shows graphically the information from Table 5.1. Here we see that the EDP metric benefits configurations that are faster, while penalizing the energy-aware ones. The FN metric has a similar slope when it comes to the fastest configurations, but does not punish slower configurations as hard.

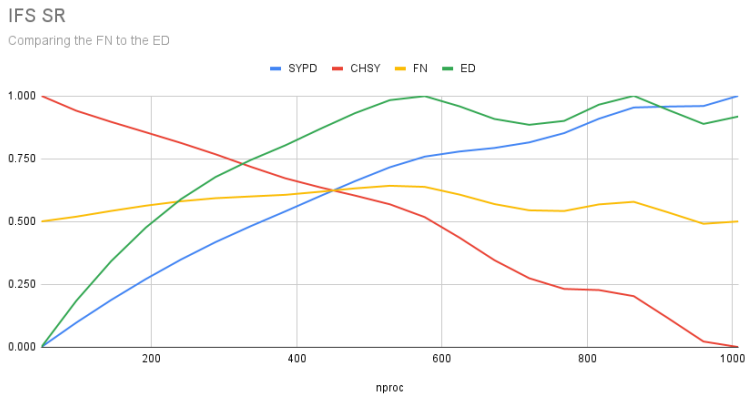


Figure 5.1: $SYPD_n$, $CHSY_n$, FN and EDP for IFS in SR. Using a TTS_r of 0.5

				FN results using different TTS_r						
nproc	$SYPD_n$	$1 - CHSY_n$	EDP	≤0.3	0.4	0.5	0.6	0.7	0.8	≥0.9
48	0.000	1.000	0.000	0.700	0.600	0.500	0.400	0.300	0.200	0.100
96	0.097	0.941	0.184	0.688	0.603	0.519	0.435	0.350	0.266	0.181
144	0.187	0.896	0.341	0.684	0.613	0.542	0.471	0.400	0.329	0.258
192	0.271	0.855	0.476	0.680	0.621	0.563	0.504	0.446	0.388	0.329
240	0.348	0.813	0.588	0.673	0.627	0.580	0.534	0.487	0.441	0.394
288	0.418	0.767	0.677	0.662	0.627	0.593	0.558	0.523	0.488	0.453
336	0.481	0.718	0.744	0.647	0.623	0.599	0.576	0.552	0.528	0.504
384	0.540	0.672	0.803	0.632	0.619	0.606	0.593	0.580	0.566	0.553
432	0.601	0.636	0.868	0.625	0.622	0.618	0.615	0.611	0.608	0.604
480	0.660	0.603	0.931	0.620	0.626	0.632	0.637	0.643	0.649	0.655
528	0.716	0.568	0.983	0.612	0.627	0.642	0.657	0.671	0.686	0.701
576	0.758	0.517	0.999	0.589	0.614	0.638	0.662	0.686	0.710	0.734
624	0.778	0.435	0.958	0.538	0.573	0.607	0.641	0.675	0.710	0.744
672	0.792	0.346	0.908	0.480	0.524	0.569	0.614	0.658	0.703	0.748
720	0.815	0.274	0.885	0.436	0.490	0.544	0.598	0.652	0.706	0.761
768	0.851	0.231	0.900	0.417	0.479	0.541	0.603	0.665	0.727	0.789
816	0.909	0.227	0.965	0.431	0.500	0.568	0.636	0.704	0.772	0.841
864	0.954	0.203	1.000	0.428	0.503	0.578	0.653	0.728	0.803	0.878
912	0.957	0.113	0.943	0.367	0.451	0.535	0.620	0.704	0.789	0.873
960	0.959	0.022	0.888	0.303	0.397	0.491	0.584	0.678	0.772	0.866
1008	1.000	0.000	0.918	0.300	0.400	0.500	0.600	0.700	0.800	0.900

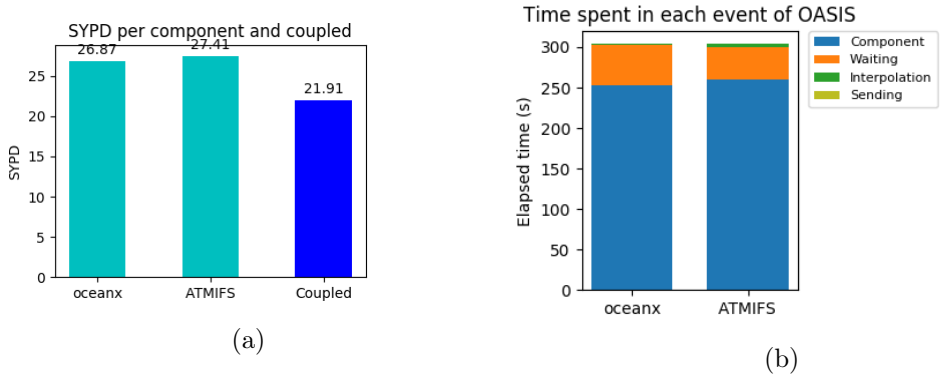
Table 5.1: Comparison between the FN and EDP metrics for IFS. Using different TTS_r values

5.2 Balancing EC-Earth Standard and High resolution experiments

As explained in Section 4.2, in the SR and HR experiments of EC-Earth, IFS and NEMO are coupled and synchronized at the beginning of their own timestep to exchange information. Ideally (and as stated in the literature), if both components run at exactly the same speed, the overhead due to the coupling would be minimal and the coupled SYPD be almost the same of its components. The waiting time would be negligible and only the extra computation due to the transformation of the data from one grid to the other (interpolation) would add some computation time to the simulation. However, Figure 5.2a shows that resource configurations in which both components run "roughly" at the same speed like using 576 processes for IFS (ATMIFS in the images) and 144 for NEMO (oceanx in the images) still performs much slower than expected. In addition, Figure 5.2b shows that the load-balance is not perfect as both components are still wasting an important amount of computing resources waiting. Figure 5.3 shows the timings for each timestep and we observe that NEMO (top) has regular timestep computations while IFS (bottom) has some irregular timesteps, which are due to the radiation code being called every 4 timesteps. This irregularity is propagated in the next timestep of NEMO as waiting time. The other 3 IFS timesteps (the regular ones) are a little bit faster than in NEMO. Therefore, IFS has to wait for NEMO in 3 out of 4 timesteps. While both components execution time is the same, the irregularities in IFS timesteps execution create an imbalance that leads to a coupled execution that is slower and wastes 16.5% (coupling cost) of the resources as waiting time. The SYPD of the coupled simulation with this configuration is 21.91. Given that it uses 720 (576+144) processes, the cost of the simulation is 789 CHSY (see Equation 5.1).

Achieving a perfect load-balance (waiting time close to 0) is impossible due to the irregular execution time of IFS timesteps. However, the solution can be improved by reallocating the resources from one component to the other. By looking at the partial coupling cost, we know the computational cost of the waiting time per component. Both components wait the same amount of time (same SYPD) but IFS uses 4 times as many processes compared to NEMO. Therefore, the computational cost of the waiting time in IFS will be much higher. Figure 5.4a shows that giving resources from IFS to NEMO (552 processes for IFS and 168 for NEMO) improves the coupled SYPD compared to the previous allocation by a 10% (24.15/21.91). As the total number of PEs is the same, the cost of the simulation (CHSY) is also improved from 789 to 715. Finally, as most of the waiting time now happens in NEMO (Figure 5.4b), which still uses less PEs, the coupling cost has also been reduced to 9.8%.

Having a set of performance metrics and tools is absolutely necessary to find this kind of solutions. The best resource configuration was only achievable by instrumenting the code (OASIS3-MCT), having a set of performance metrics (CPMIP) and manually testing multiple resource allocations for each model. The optimization shown above required to run the scalability of IFS and NEMO separately, find a number of processors so that both have the same execution time, get the performance metrics for a coupled experiment with that resource configuration, realize that it would be better to have NEMO running faster to reduce the coupling cost and then try multiple tests where a subset of IFS processes are reallocated to NEMO, until the best solution is found.



(a) (b)

Figure 5.2: Coupled execution running IFS and NEMO at the same speed. Figure 5.2a shows the SYPD of IFS, NEMO and for the Coupled simulation. Figure 5.2b shows the time spent executing component operations and on events related to the coupling (waiting, interpolation, sending).

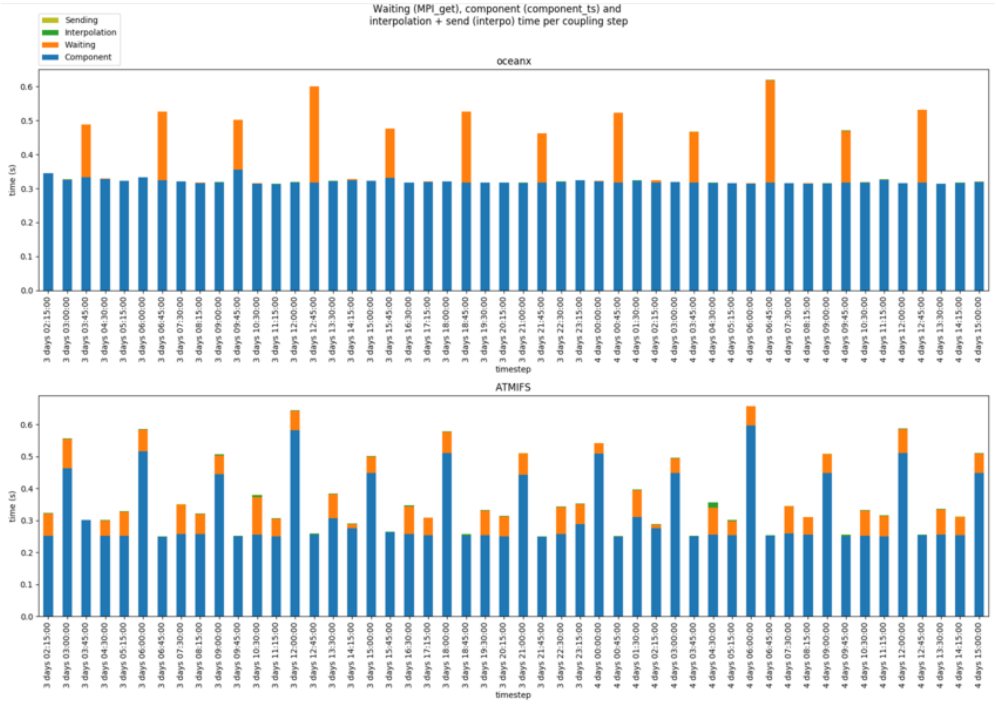


Figure 5.3: Information per timestep for IFS-NEMO coupled executions when both components run at the same speed.

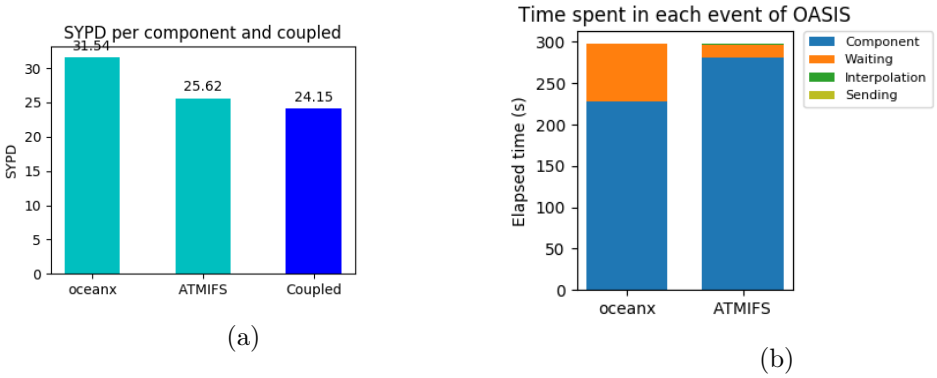


Figure 5.4: Coupled execution running IFS and NEMO in a balanced resource configuration. Figure 5.4a shows the SYPD of IFS, NEMO and for the Coupled simulation. Figure 5.4b shows the time spent executing component operations and on events related to the coupling (waiting, interpolation, sending).

Chapter 6

Automatic load-balance method

The following section describes the automatic load-balance method (auto-lb), which can be divided in 3 main steps:

1. Scalability workflow: Get the SYPD (i.e. execution time) for each one of the components involved in the coupled configuration under different processor counts. There is no strict rule on how many PEs counts should be tested or how many runs for each PEs count may have to be executed, as it depends on the component under analysis and the possible variability of the machine. Having more PEs counts tested and running each one for longer (or more than once) will result in a better representation of the performance of the model (at the expense of greater computing power required).

2. Prediction script: A Python script that, given the scalability curves of the components involved in the coupled configuration, will return the best NP allocation depending on the user criteria (TTS or ETS) using the FN metric.

3. Load-balance workflow: Workflow controlled by Autosubmit that will submit the jobs on the HPC machine with the NP allocation that the Prediction script estimate as optimal. Validating the results and doing some fine-grain modifications to achieve a better load-balance.

6.1 Scalability workflow

The load-balance method can only modify the number of PEs allocated for each of the components to obtain a better resource configuration. Thus, it is essential to know how each of the models involved in the coupled configuration performs at different processor counts. Until now, for all the EC-Earth community around Europe, there wasn't any way of getting this information without having to run each of the components in standalone for multiple processor counts manually. A new workflow has been created to automatize this process. Moreover, it also optimizes the work by allowing to use coupled executions to get a point of the scalability curve (nproc,SYPD) for each of the models involved in one run. This is done using the OASIS3-MCT internal load-balance tool (LUCIA). It collects the timings of all coupled events within a component during the runtime. The time outside the coupler is, therefore, the time a component spends in its own computations and it is equivalent to its execution in stand-alone. Hence, running a single instance of an IFS-NEMO experiment will now suffice to get the SYPD of IFS and NEMO (at their respective number of PEs). Autosubmit is used to automatize the process. Which follow the next steps:

1. Configure the coupled experiment under study.

2. Define a list with the number of PEs to use for each of the components in a text file (nproc_config). For instance: IFS_nproc=(48 144 384 480 576 684 768 912) and NEMO_nproc=(48 96 144 192 240 288 336 384) will run 8 members (different realizations). The first one with 48 processes for both components, the second with 144 for IFS and 96 for NEMO, and son on.

3. In the Autosubmit configuration, define the number of chunks (individual executions) and their duration. Running multiple chunks for longer periods helps to mitigate the variability of the HPC machine.

4. Run a bash script to create the Autosubmit jobs and their dependencies. This script takes the list of PEs from the nproc_config file and modifies the jobs.conf file that AS uses to create the workflow.

5. POST_LUCIA job: New Autosubmit job (bash script) to get the performance metrics of a chunk. Used to get the SYPD of IFS and NEMO running with a particular number or PEs.

6. SCALABILITY job: New Autosubmit job to transfer the performance results form the HPC machine to the local workstation and do the average result of the executed chunks.

Figure 6.1 is an example of the scalability workflow for 8 different PE allocations for IFS and NEMO. Each resource configuration will be executed twice (2 chunks). The POST_LUCIA job runs at the end of each member to get the performance metrics for all the executed chunks. Finally, all the jobs are synchronized with the SCALABILITY job, which transfers and gathers all the performance metrics in the local machine, does the average of the chunks executed in the same member and creates the CSV files with the nproc,SYPD points for each component.

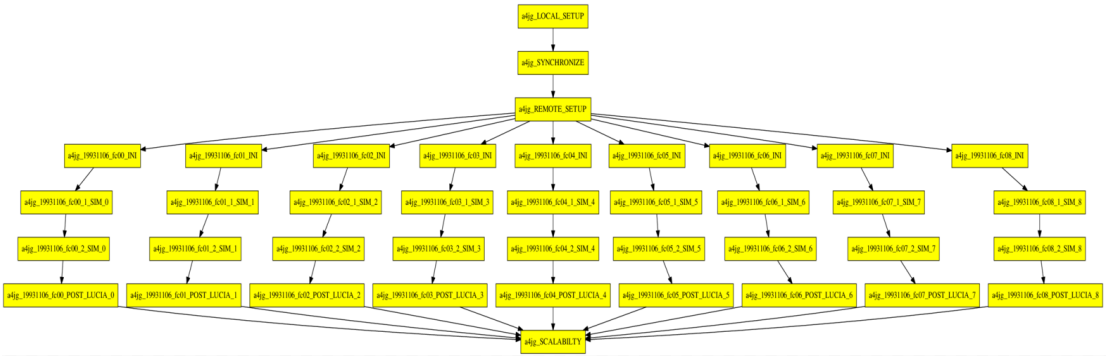


Figure 6.1: Auto-scalability workflow for 8 different PE allocations per component (members) running 2 chunks each.

6.2 Prediction script

The number of possible configurations that can be used in coupled ESMs is so large that it would not be practical to test all of them one by one. For instance, in IFS-NEMO experiments where both components can easily take any number of PEs from 48 (1 node)

to 1008 (21 nodes), the number of solutions using a granularity of say, 24 processes (half a node), is $20/0.5 * 20/0.5 = 1600$. Most of them, however, are configurations that would be very imbalanced in practice and testing them would only be a waste of resources. Nevertheless, the Prediction script can search in this solution space in less than one second and approximate the results from each combination of PEs for IFS-NEMO based on the prior knowledge of the parallel behaviour that we have from the scalability curves. The Prediction script is a Python script configured through a YAML file where the user can set the next parameters:

- **max_nproc**: Add a limitation on the maximum number of PEs to use for the prediction.
- **TTS_ratio**: Float number between 0 and 1. Note that the ETS_r is inversely proportional as described in Equation 5.7
- **interpo_method**: Choose the interpolation method to use for the scalability curves (linar, slinear, quadratic, cubic).
- **show_plots**: Boolean. Show some extra information as plots. Used for debug proposes.
- **node_size**: Define the number of PEs that each node has. Machine dependent value.

And also the settings for each component:

- **Component name**: A string with the name to give to the component
- **scalability_file**: Path to the scalability curve, a CSV file with the PE count in the first column and the corresponding SYPD in the second.
- **nproc_restriction**: Array with the only PEs that this component can take. In the format [48,128,..,912].
- **timestep_info** (optional): Path to the information per timestep. A CSV file with the time spent in component execution, waiting, interpolating and sending at each timestep.
- **timestep_nproc** (mandatory only if timestep_info is set): The number of resources used by the component when the timestep_info timers were collected.

If the information per timestep is not present, the script will by default assume that the models have perfect regular timestep lengths. When provided, it allows the script to better understand the coupling phase and estimate more accurately the computational cost it will add.

The Prediction script will return the top 5 best configurations for the TTS/ETS criteria selected while respecting the limitations (max_nproc, nproc_restriction) specified by the user. It also estimates what the SYPD, CHSY and coupling cost will be for the best solution found. TextBox 6.1 shows an example of the output of the Prediction script. Starting from the top, it shows the parameters set by the user. Then the information per component (number of processes, SYPD and CHSY). Finally, the coupled solution that expects to be optimal, with the total number of processes used, the coupled SYPD and CHSY, the coupling_cost and the ratio of the execution time of both components.

The process of interpolating the scalability curve, exploring all the possible combinations of PEs and finding the best ones is explained in the following sections.

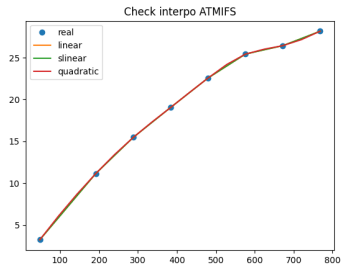
```
Using a limitation of 2000 processes at most.
No information per timestep available.
Assuming regular timestep lengths
Optimal for TTS=0.5, ETS=0.5:

-----------------------------------------------------

Results for component ATMIFS:
Number of processes: 720
CHSY: 5112
SYPD: 3.38

-----------------------------------------------------

Results for component oceanx:
Number of processes: 1245
CHSY: 9054
SYPD: 3.30

-----------------------------------------------------

Total number of processes: 1965
Expected coupled CHSY: 14290
Expected coupled SYPD: 3.30
Expected coupling cost: 0.87 %, (123.94 CHSY)
ATMIFS/oceanx speed ratio: 1.02
```

TextBox 6.1: Prediction script output example.

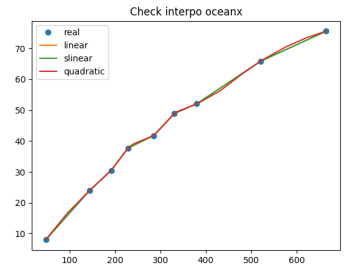
6.2.1 Interpolate the scalability curves

Running the scalability tests for a model using all the possible processor counts that it can take would be just a waste of computational resources. Instead, one can approximate the result of a particular processor count by looking at its nearest points and expecting to be a value between both, given that the execution time of a model can be seen as a nonlinear function of the process counts it uses. In the Prediction script, the scalability curves of each component are interpolated (using the *interpolate*[1] function from the scipy python package) to have the expected SYPD when using a number of PEs different than the points (nproc,SYPD) from the original file. If nproc_restriction is set for a component, it will only interpolate for those PEs. If not, the node_size is used instead. There are different algorithms to interpolate the scalability curves (e.g. linear, slinear, quadratic, cubic...). But the prediction values do not change too much, as it is shown in Figure 6.2. The interpolation can be used to estimate the scalability values for any processor count without having to run the simulation. For instance, it is used in Table 6.3 to have the SYPD of IFS and NEMO at processor counts multiple of the node size (48 cores).

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.interp1d.html



(a)



(b)

Figure 6.2: Using different interpolation methods to extend the scalability curves and be able to predict the SYPD using any processor count. Figure 6.2a compares the real scalability values for IFS against various interpolation algorithms and Figure 6.2b does the same for NEMO

nproc	SYPD
48	3.27
96	5.92
144	8.41
192	10.76
240	12.96
288	15.01
336	16.64
384	17.34
432	18.25
480	20.27
528	21.37
576	20.81

Table 6.1

nproc	SYPD
48	3.53
96	7.70
144	11.84
192	15.92
240	19.65
288	23.03
336	26.37
384	29.62
432	32.76
480	35.45
528	37.38
576	38.57

Table 6.2

Table 6.3: Scalability values after interpolating for IFS (Figure 6.2a) and NEMO (Figure 6.2b)

6.2.2 Create the TTS and ETS matrices

Once the SYPD have been interpolated for both components at the required PE counts (Table 6.3), the TTS matrix (TTS_{mx}) is build by speculating that the coupled SYPD will be the minimum of both components. Table 6.4 shows the minimum SYPD achieved for all possible IFS and NEMO nproc combinations.

The CHSY and SYPD are related with Equation 5.1. Therefore, from the TTS_{mx} we can get the ETS_{mx} (Table 6.5). Likewise, the EDP of all possible nproc combinations (EDP_{mx}) is shown in Table 6.6 applying Equation 5.6.

Most of the configurations are really bad, mainly the ones in the first columns (rows) where IFS (NEMO) is kept at very few PEs and we start giving more resources to NEMO (IFS). The coupled SYPD stays at the same as the component running with few resources, but the number of processes allocated grows. Therefore, the CHSY starts increasing a lot, as it is seen in the ETS_{mx}. To create the fitness matrix (FN_{mx}) we first have to normalize the SYPD (TTS_{mx}) and the CHSY (ETS_{mx}), and having values that are too far apart caused difficulties when trying to compare configurations that

were very close to each other. Table 6.7 is an example of how the FN_{mx} would look like if all combinations are taken into consideration. We observe that the FN values for the best candidates are very close to each other. After normalizing, small differences (below ∼50 CHSY) are hidden by having extreme values (over 4000 CHSY) in configurations like 48 PEs for IFS and 576 for NEMO. Looking at the EDP_{mx} in Table 6.6, we see that the worst resource configurations have an EDP lower than the base-case (in dark-gray). And of course, they will never be selected as the best configurations. At the same time, the best configurations (in green) are found in a region in which the predicted CHSY and SYPD are quite similar. We enforce the final solution to have an EDP better than the base-case to remove the extreme values and be able to highlight small differences between good configurations. The result of this filtering in shown in Table 6.8.

IFS nprocs \ NEMO nprocs	48	96	144	192	240	288	336	384	432	480	528	576
48	3.27	3.27	3.27	3.27	3.27	3.27	3.27	3.27	3.27	3.27	3.27	3.27
96	3.53	5.92	5.92	5.92	5.92	5.92	5.92	5.92	5.92	5.92	5.92	5.92
144	3.53	7.7	8.41	8.41	8.41	8.41	8.41	8.41	8.41	8.41	8.41	8.41
192	3.53	7.7	10.76	10.76	10.76	10.76	10.76	10.76	10.76	10.76	10.76	10.76
240	3.53	7.7	11.84	12.96	12.96	12.96	12.96	12.96	12.96	12.96	12.96	12.96
288	3.53	7.7	11.84	15.01	15.01	15.01	15.01	15.01	15.01	15.01	15.01	15.01
336	3.53	7.7	11.84	15.92	16.64	16.64	16.64	16.64	16.64	16.64	16.64	16.64
384	3.53	7.7	11.84	15.92	17.34	17.34	17.34	17.34	17.34	17.34	17.34	17.34
432	3.53	7.7	11.84	15.92	18.25	18.25	18.25	18.25	18.25	18.25	18.25	18.25
480	3.53	7.7	11.84	15.92	19.65	20.27	20.27	20.27	20.27	20.27	20.27	20.27
528	3.53	7.7	11.84	15.92	19.65	21.37	21.37	21.37	21.37	21.37	21.37	21.37
576	3.53	7.7	11.84	15.92	19.65	20.81	20.81	20.81	20.81	20.81	20.81	20.81

Table 6.4: TTS matrix for IFS-NEMO coupled execution. IFS PEs in the vertical axis. NEMO PEs in the horizontal

IFS nprocs \ NEMO nprocs	48	96	144	192	240	288	336	384	432	480	528	576
48	705	1057	1409	1761	2114	2466	2818	3171	3523	3875	4228	4580
96	979	778	973	1168	1362	1557	1751	1946	2141	2335	2530	2724
144	1305	748	822	959	1096	1233	1370	1507	1644	1781	1918	2055
192	1632	898	749	857	964	1071	1178	1285	1392	1499	1606	1713
240	1958	1047	778	800	889	978	1067	1156	1244	1333	1422	1511
288	2284	1197	876	767	844	921	998	1074	1151	1228	1305	1381
336	2611	1346	973	796	831	900	969	1038	1108	1177	1246	1315
384	2937	1496	1070	868	864	930	997	1063	1129	1196	1262	1329
432	3263	1646	1168	941	884	947	1010	1073	1136	1199	1262	1326
480	3590	1795	1265	1013	879	909	966	1023	1080	1137	1193	1250
528	3916	1945	1362	1085	938	916	970	1024	1078	1132	1186	1240
576	4242	2095	1459	1158	997	996	1052	1107	1163	1218	1273	1329

Table 6.5: ETS matrix for IFS-NEMO coupled execution. IFS PEs in the vertical axis. NEMO PEs in the horizontal

IFS nprocs \ NEMO nprocs	48	96	144	192	240	288	336	384	432	480	528	576
48	1.000	0.667	0.500	0.400	0.333	0.286	0.250	0.222	0.200	0.182	0.167	0.154
96	0.777	1.639	1.311	1.093	0.936	0.819	0.728	0.656	0.596	0.546	0.504	0.468
144	0.583	2.218	2.205	1.890	1.654	1.470	1.323	1.203	1.102	1.018	0.945	0.882
192	0.466	1.848	3.094	2.707	2.406	2.166	1.969	1.805	1.666	1.547	1.444	1.353
240	0.388	1.584	3.278	3.491	3.142	2.856	2.618	2.417	2.244	2.094	1.963	1.848
288	0.333	1.386	2.913	4.214	3.831	3.512	3.242	3.010	2.809	2.634	2.479	2.341
336	0.291	1.232	2.622	4.310	4.316	3.984	3.699	3.453	3.237	3.046	2.877	2.726
384	0.259	1.109	2.384	3.950	4.326	4.017	3.749	3.515	3.308	3.124	2.960	2.812
432	0.233	1.008	2.185	3.647	4.450	4.153	3.894	3.664	3.461	3.279	3.115	2.966
480	0.212	0.924	2.017	3.386	4.815	4.803	4.521	4.269	4.045	3.842	3.660	3.493
528	0.194	0.853	1.873	3.160	4.514	5.025	4.745	4.496	4.271	4.067	3.883	3.714
576	0.179	0.792	1.748	2.963	4.248	4.500	4.263	4.050	3.857	3.682	3.522	3.375

Table 6.6: EDP matrix for IFS-NEMO coupled execution. IFS PEs in the vertical axis. NEMO PEs in the horizontal

IFS nprocs \ NEMO nprocs	48	96	144	192	240	288	336	384	432	480	528	576
48	0.5	0.45	0.41	0.36	0.32	0.27	0.23	0.18	0.14	0.09	0.05	0
96	0.47	0.56	0.54	0.51	0.49	0.46	0.44	0.41	0.39	0.36	0.34	0.31
144	0.43	0.62	0.63	0.61	0.59	0.57	0.56	0.54	0.52	0.5	0.49	0.47
192	0.39	0.6	0.7	0.69	0.67	0.66	0.65	0.63	0.62	0.6	0.59	0.58
240	0.35	0.58	0.73	0.76	0.74	0.73	0.72	0.71	0.7	0.69	0.68	0.66
288	0.3	0.56	0.71	0.82	0.81	0.8	0.79	0.78	0.77	0.76	0.75	0.74
336	0.26	0.54	0.7	0.84	0.85	0.84	0.84	0.83	0.82	0.81	0.8	0.79
384	0.22	0.52	0.69	0.83	0.87	0.86	0.85	0.84	0.83	0.83	0.82	0.81
432	0.18	0.5	0.68	0.82	0.89	0.88	0.87	0.87	0.86	0.85	0.84	0.83
480	0.13	0.48	0.66	0.81	0.93	0.94	0.94	0.93	0.92	0.91	0.91	0.9
528	0.09	0.46	0.65	0.8	0.92	0.97	0.97	0.96	0.95	0.94	0.94	0.93
576	0.05	0.44	0.64	0.79	0.91	0.95	0.94	0.93	0.93	0.92	0.91	0.9

Table 6.7: Fitness matrix for IFS-NEMO coupled execution using a TTS_r of 0.5 and without filtering by the EDP. IFS PEs in the vertical axis. NEMO PEs in the horizontal

6.2.3 Building the final solution

Table 6.8 shows the Fitness metric using a TTS_r of 0.5. The optimal value is the same as the one found by the EDP metric (Table 6.6), which is using 528 processes for IFS and 288 for NEMO. However, these values represent only predicted results based on the scalability curves provided to the script and one expects to have some variability when running real experiments on the supercomputing machine. Consequently, the Prediction script will not only return the best solution but the best 5 (top5). In the example above, that is 528-288, 528-336, 480-240, 480-288 and 480-336 (IFS-NEMO).

The Prediction script will, therefore, serve as a guide for the Load-balance workflow as now it will search in a smaller space to find the best resource configuration without having to waste as many resources testing combinations of PEs for IFS and NEMO that can not be an optimal solution for the ESM.

IFS nprocs \ NEMO nprocs	48	96	144	192	240	288	336	384	432	480	528	576
48	0.5	-	-	-	-	-	-	-	-	-	-	-
96	-	0.54	0.45	0.36	-	-	-	-	-	-	-	-
144	-	0.6	0.59	0.52	0.46	0.4	0.33	0.27	0.21	0.14	-	-
192	-	0.53	0.69	0.64	0.59	0.54	0.49	0.44	0.39	0.34	0.29	0.24
240	-	0.46	0.7	0.72	0.68	0.64	0.6	0.56	0.52	0.48	0.43	0.39
288	-	0.39	0.66	0.8	0.76	0.72	0.69	0.65	0.62	0.58	0.55	0.51
336	-	0.32	0.61	0.81	0.81	0.78	0.75	0.71	0.68	0.65	0.62	0.59
384	-	0.25	0.57	0.77	0.81	0.78	0.75	0.72	0.69	0.66	0.63	0.6
432	-	0.19	0.52	0.74	0.83	0.8	0.77	0.74	0.71	0.68	0.65	0.63
480	-	-	0.48	0.71	0.87	0.87	0.85	0.82	0.8	0.77	0.74	0.72
528	-	-	0.43	0.67	0.84	0.9	0.88	0.85	0.83	0.8	0.78	0.75
576	-	-	0.39	0.64	0.82	0.85	0.82	0.8	0.77	0.75	0.72	0.69

Table 6.8: Fitness matrix for IFS-NEMO coupled execution using a TTS_r of 0.5. IFS PEs in the vertical axis. NEMO PEs in the horizontal.

6.3 Load-balance workflow

The last part of the method is the load-balance workflow. This is an iterative process that will submit the top5 best configurations found by the Prediction script, run multiple chunks for each resource configuration to lower the impact that the HPC platform variability can add, do some small improvements (changing the number of PEs from one component to the other) based on the performance results from real executions and propose a new set of 5 resource configurations. The process will converge when no new configurations can be explored. Once that point is reached, it will calculate the Fitness metric for all the PEs combinations tested and return the best resource configuration (the one that maximizes the FN).

6.3.1 Creating the list of jobs and their dependencies

Firstly, the workflow jobs and dependencies have to be created given the top5 best configurations coming from the Prediction script. The procedure is similar to what was done in Section 6.1. It consists of a bash script (create_lb_jobs.sh) that takes the top5 resource configurations and the target jobs.conf files as parameters. Additionally, the user has to specify the total number of iterations that the workflow will run (lb-iterations). This is not known beforehand, but since AS does not support the creation of jobs dynamically a maximum value must be defined. There is no penalization on adding more lb-iterations than needed as once the method has converged, the workflow will stop and jobs that were waiting will not be executed. Normally, setting the lb-iterations to 8 is enough. But it depends on the step size used (number of processes that are given from one component to the other) and on how good the initial resource configurations were (a configuration with a high load imbalance will take more time to achieve a balanced configuration). Figure 6.3 is an example of the workflow that will run 2 chunks per resource configuration.

6.3.2 POST_LUCIA job

A new job (bash script) to get the performance metrics needed for the load-balance optimization. It executes the LUCIA tool, which provides the CPMIP metrics from all

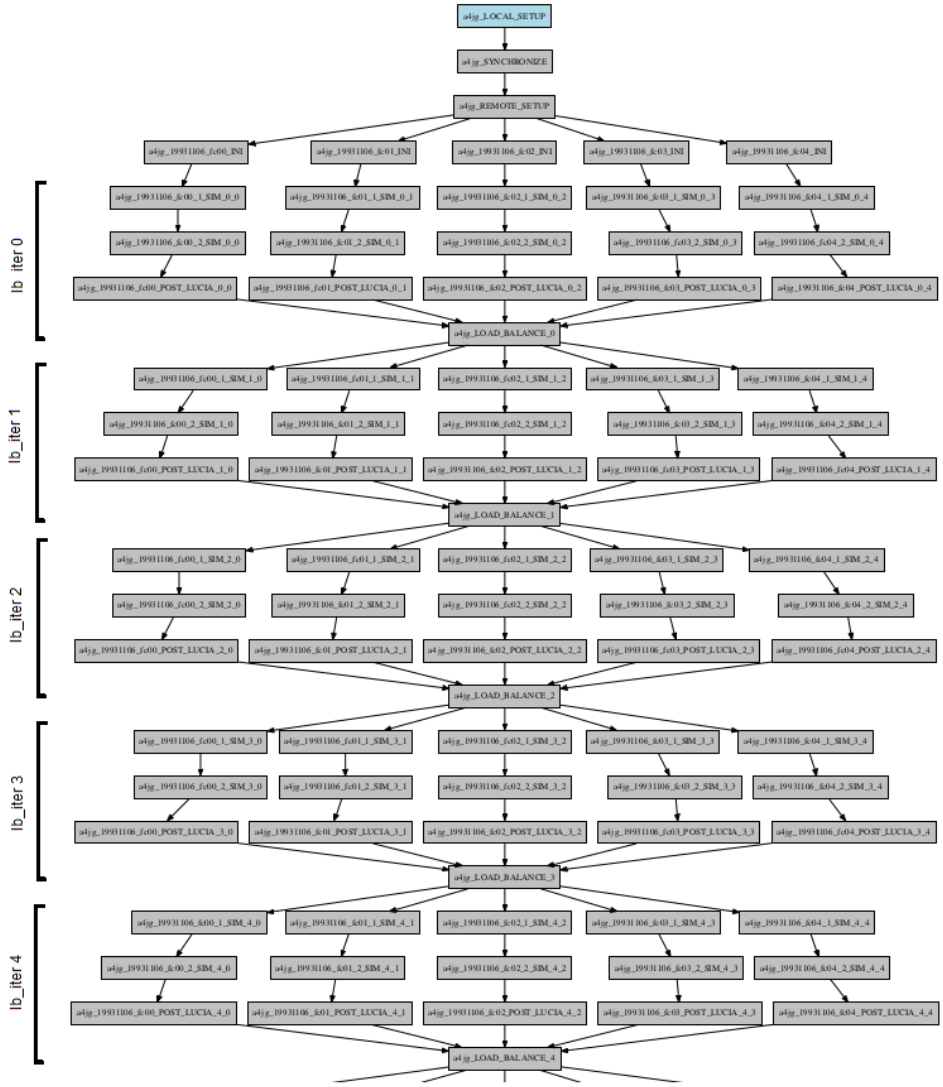


Figure 6.3: Overview of the load-balance workflow. An iterative process that submits simulation jobs to the HPC machine for different resource configurations (SIM), gets the performance results (LUCIA) and give resources from one component to the other to achieve better balanced NP allocations (LOAD_BALANCE)

the chunks executed by a member.

6.3.3 LOAD_BALANCE job

A new job to control which component will give PEs (donor) and which will receive them (recipient) in the next iteration. First, it will transfer the results from the HPC machine to the local platform. Once all the performance metrics are in the local platform, the LOAD_BALANCE job will read them and use the Partial_coupling_cost (Equation 5.3) to determine which component adds more to the coupling cost. That component will be tagged as the donor, as it is wasting more resources and/or for longer time. The other component will be the recipient. The number of PEs to reallocate at once is the step size and can be any number. Likewise, the job defines a minimum step size to avoid doing reallocations of PEs counts that are too small to have any impact on the execution and improve the solution.

By default, the step size is the same as the size of a node but it could depend on the models and the initial resource configurations used. Components that do not experiment rapid changes in their execution time when adding or reducing the step size of 1 node should use bigger step sizes. On the other side, if a small modification on the number of PEs results in significant changes, it should be better to use smaller step sizes. Finally, if the resource configuration is already a balanced one, moving many PEs in one step between components will make the solution worse. But if the resource configuration is very imbalanced, then having a bigger step will help to find a balanced one with less intermediate steps.

Figure 6.4 is an example of how the LOAD_BALANCE job modifies the number of processes of IFS and NEMO to achieve more balanced configurations. The initial configurations (the ones in the first iteration or lb-iter 0) correspond to, left to right:

- Predicted: the best prediction configuration found by the Prediction script using a $TTS_r = 0.5$, 528 IFS - 192 NEMO (test 0)
- same SYPD: a configuration where IFS and NEMO run at the same SYPD, 610 IFS - 148 NEMO (test 1)
- EDP: the best resource configuration using the EDP metric, 576 IFS - 168 NEMO (test 2)
- same nproc: a configurataion where IFS and NEMO use the same number of PEs, 384 IFS - 380 NEMO (test 3)

After the first iteration, the number of PEs exchanged between components will be the one defined by the step size variable. Therefore, all tests in lb-iter 1 are the result of giving 48 processes from IFS to NEMO (tests 1 and 2) or from NEMO to IFS (tests 0 and 3) depending on the performance results (i.e. the Partial coupling cost) from the previous lb-iter (lb-iter 0) for each test.

Before defining the resource configurations to use in the next lb-iter, the LOAD_BALANCE job checks that it will indeed be a new configuration (not executed by any of the tests in any of the lb-iters before). If the candidate resource configuration is not a new one, the step size for that test is divided by 2. This is illustrated in the evolution of test 0: lb-iter 1 resource configuration is the result of modifying the allocation used in lb-iter 0 by giving 48 PEs from NEMO (192 -> 144) to IFS (528 -> 576), and lb-iter 2 is the result of changing lb-iter 1 by giving 24 PEs from IFS (576 -> 552) to NEMO (144 -> 168). If the step was not reduced the resource configuration of lb-iter 2 would be the same as the initial one. If the LOAD_BALANCE job can not find a new resource configuration for a test without using a step size smaller than the minimum allowed, that test will not be executed anymore. In the example, test 3 is the one that takes longer to converge, making tests 0 and 2 wait (i.e. the number of PEs for IFS and NEMO and the performance results are the same) for lb-iterations 5 to 7 and test 1 for lb-iterations 6 and 7.

Figure 6.5 shows how the coupling improves for different initial resource configurations. Figure 6.6 shows the same but for the Fitness values. At the first lb-iteration, the "Predicted" (blue) is the configuration with the highest FN and the lowest coupling cost. But in the next iteration (lb-iter 1), the step size is too big and the configuration gets worse (the coupling cost is doubled). Subsequent lb-iterations keep improving the Fitness (and coupling cost) until lb-iteration 4. The configuration using the EDP metric (yellow) is a bit inferior at the beginning (lb-iteration 0) but it is improved in the next iteration. During the lb-iteration 2 it becomes worse than with its initial setup, but

this test is needed to then find its better configurations in lb-iterations 3 and 4. The approach of using the same SYPD (red) for IFS and NEMO was already discussed in Section 5.2. This example shows again how this configuration is suboptimal. In the first lb-iteration the coupling cost is 20% and the FN is less than the Predicted and EDP configurations. But the auto-lb workflow can balance this configuration in subsequent iterations to reduce the coupling cost and increase its FN. Finally, the worst configurations is the same nproc (green). It starts having a coupling cost over 30% and it is the one with the smallest initial FN. But next lb-iterations can reduce the coupling cost and make it a better resource configuration.

Notice that starting the auto-lb with a resource configuration that is imbalanced (e.g. same nproc) makes the workflow converge much slower than when starting with a well balanced one (e.g. Predict). We can observe this in Figure 6.5: the "Predicted" resource configuration takes 5 lb-iterations to converge and the "same nproc" takes 8. Therefore, while the lb-workflow can balance any resource configuration, providing a well balanced one is important as it reduces the amount of lb-iterations needed to finish (i.e. the amount of simulations to run on the HPC platform and the overall cost of running the workflow).

The workflow finishes if the LOAD_BALANCE job can not find any new resource configuration for the next lb-iter. Later, it will create the plot with the performance metrics of all the executed tests and mark the best one in green (as seen in Figure 6.4).

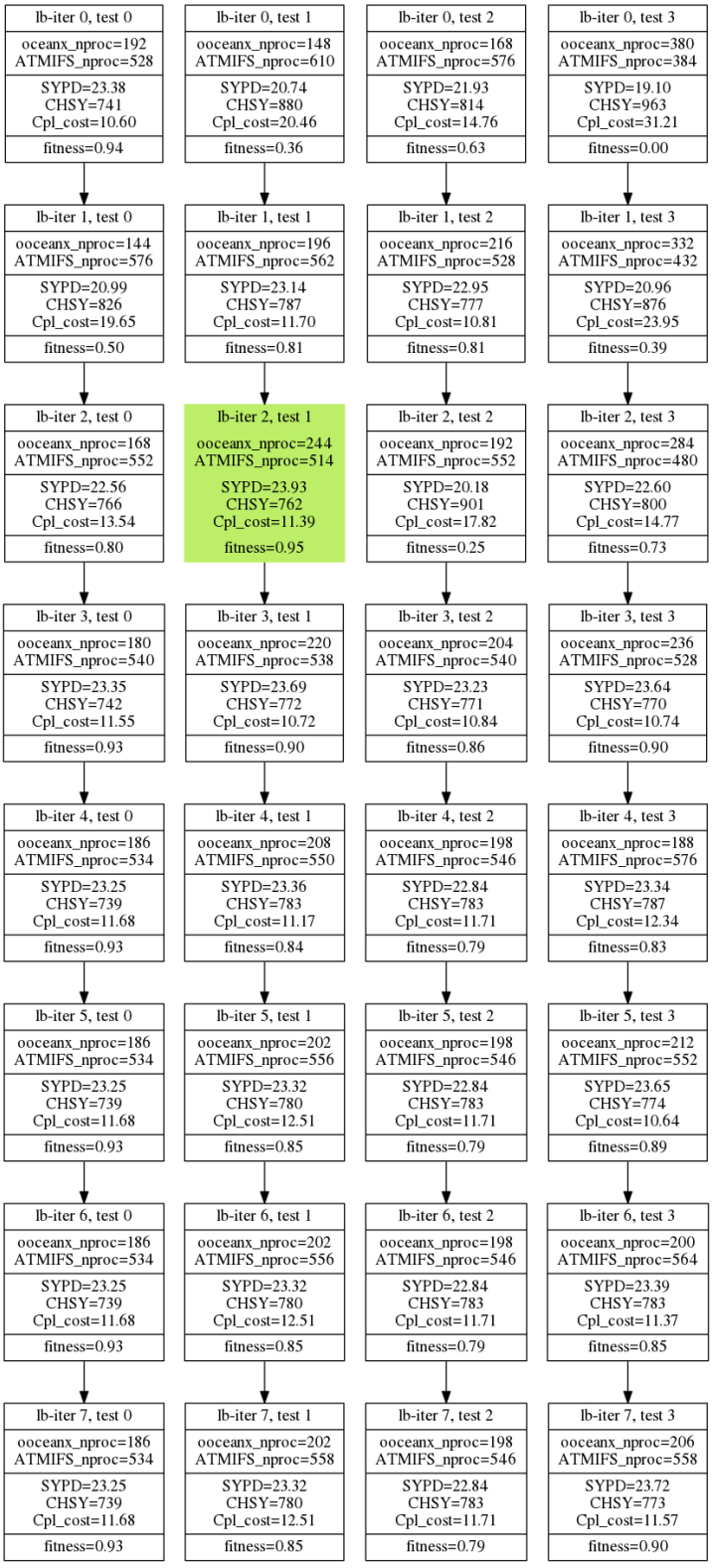


Figure 6.4: Example illustrating the Load-balance workflow.

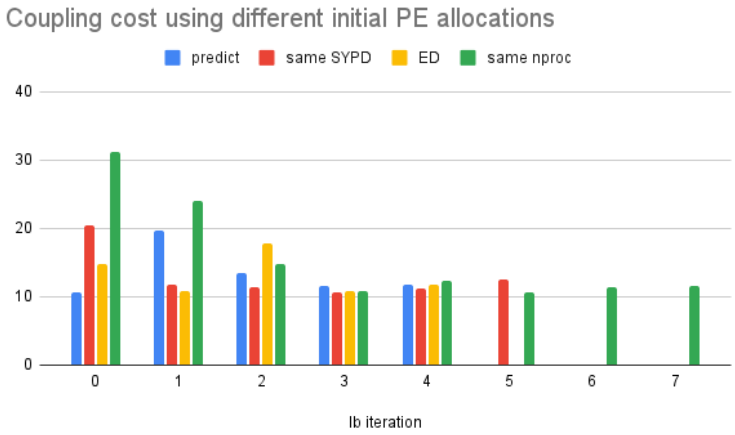


Figure 6.5: The coupling cost is reduced by allocating PEs from one component to the other.

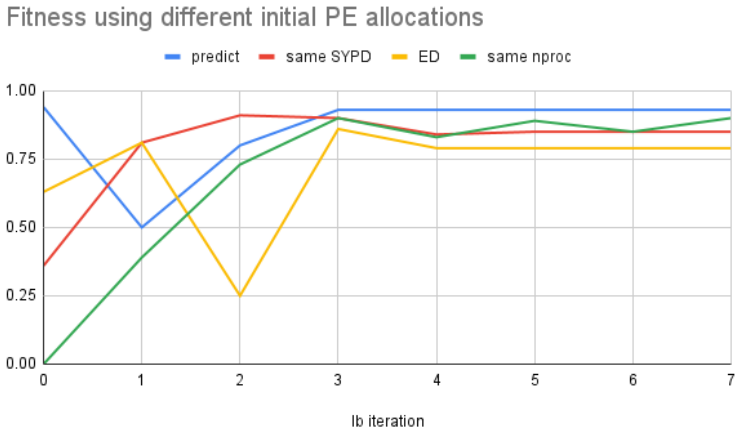


Figure 6.6: The FN value tends to improve as the workflow iterates. But the best solution does not have to be at the last one.

6.4 Portability to ECMWF HPC

EC-Earth3 was already deployed on the ECMWF supercomputer. Therefore, the porting of EC-Earth3 was already done and using the AS workflow manager to handle the tasks. So no further changes were needed in that regard. However, some modifications to the auto-lb method had to be introduced to run on this machine. Firstly, the scheduler used in ECMWF machine is LSF, while SLURM is used in MareNostrum. That means that the header of the batch jobs had to be modified on the script that creates the job list for AS (Section 6.3.1), translating the SLURM commands to LSF. LSF requires defining explicitly the allocation of each MPI process to a physical core. Table 6.9 shows the differences when running an experiment with 144 processors divided as 1 XIOS, 36 NEMO, 106 IFS and 1 RNF. In SLURM only the total amount of processes has to be defined, in LSF the "tasks" command is used to map the number of PEs to use each component. Additionally, the "tasks-per-node" command (which in SLURM is set to be the size of a node) has to be set to the number of MPI processes to run on a node for each one of the binaries. Finally, the "export" command used in SLURM to set-up

command	SLURM	LSF
tasks	144	1:36:106:1
tasks-per-node	-	1:36:36:1
export	#SBATCH –export=ifs_np=106, nemo_np=36	#PBS -v ifs_np=106, nemo_np=36

Table 6.9: Summary of scheduler command modifications to port the auto-lb method from MareNostrum (SLURM) to ECMWF (LSF)

	MN	ECMWF
Processor type	Intel Xeon Platinum 8160 CPU	Intel E5-2695v4 “Broadwell”
Cores per CPU	24	18
CPUs per node	2	2
Cores per node	48	36
Memory per node (GB)	96	128
Network	100 Gbit/s Intel Omni-Path	16 Gbit/s Cray Aries
Linpak Rmax (PFlop/s)	6.47	3.94

Table 6.10: MareNostrum4 and ECMWF HPC machine overview

the IFS and NEMO processes to for a particular run was changed to the equivalent "-v" command in LSF.

Most part of the POST_LUCIA job (Section 6.3.2) could remain as it was for MareNostrum. The only change was to load the required Python version and libraries before running the LUCIA tool and get the performance metrics of the simulations. The LOAD_BALANCE job (Section 6.3.3) had to be modified to handle the transferring of files from the ECMWF to the local machine. A "data_transfer" node is used as a bridge to get the performance metrics as direct connections were not allowed. The performance results are first moved to the "data_transfer" node and then transferred to the local machine running AS. If this job finishes correctly, it will keep only the local files. Thus, it deletes all the copies from the ECMWF machine and "data_transfer" node.

Table 6.10 shows some of the differences between the two HPC machines. The aim of this work is not to compare how the same model performs on different platforms but to highlight that the auto-lb method is capable of finding optimal solutions across varying hardware, probing the portability of the new approach. The size of a node (in number of cores) is 48 in MareNostrum4 and 36 in ECMWF. This value is used by the Prediction script as the granularity to interpolate the scalability curves and by the Load-balance workflow to set the starting step (i.e. the number of processes reallocated from one component to the other to achieve a better load-balance). This influences the resource configurations that will be tested and the number of lb-iterations executed before the workflow stops.

Chapter 7

Results

The auto-lb method has been used to optimize the resource configuration for IFS-NEMO executions in SR and HR in two different platforms. Firstly, the method is used on the ECMWF HPC to evaluate the performance gains against processor allocations that were recommended by only looking at the scalability properties of both components. Later, the auto-lb method is contrasted against configurations that were specially designed for very long experiments in CMIP6 and EUCP projects. In these two configurations the resource allocations were carefully chosen to be optimal after manually trying multiple combinations.

For each of the following performance studies we have used the auto-scalability workflow (Section 6.1) to get the speedup for each of the components, executed the Prediction script (Section 6.2) to interpolate the curves to have the SYPD values at PEs multiple of a node size and to generate the top 5 best predicted resource configurations, and run the Load-balance workflow (Section 6.3) to find the optimal processors allocation.

7.1 ECMF

7.1.1 High resolution experiment in ECMWF machine

The scalability of IFS (ATMIFS in the images) and NEMO (oceanx in the images) using the HR grids on ECWMF is shown in Figure 7.1. IFS scalability (Figure 7.1a) shows that its speed can take a range between 0.5 and 3.5 SYPD depending on the number of PEs used. The cost increases slightly as we use more processors, being 5000 CHSY the minimum and almost 9000 the maximum. There are two regions where the model seems to have an important parallel efficiency loss: after 800 and over 1080 PEs. On the other hand, NEMO (Figure 7.1b) shows a linear speedup as the cost does not increase as we add more resources, but stays close to 9000 CHSY. The SYPD it rages between 0.25 and 4.

Table 7.1 shows the top5 best predicted resource configurations using a TTS_r of 0.5, without any limitation on the maximum number of resources and using a step of one node (36 processes) for the prediction script. Additionally, the last row corresponds to the recommended set-up. The best predicted resource configuration is also shown in Figure 7.1, giving 1080 PEs to IFS and 1343 to NEMO.

Figure 7.2 shows the different resource configurations tested by the Load_balance workflow with the performance metrics averaged after running a simulation of 2 months twice. The original configuration gives a SYPD of 1.94 using 1404 processes (504 IFS, 900 NEMO), and the CHSY is 17709. The cost due to the load-imbalance is 17.02%. The optimal solution found by the workflow is one of the recommended by the Prediction

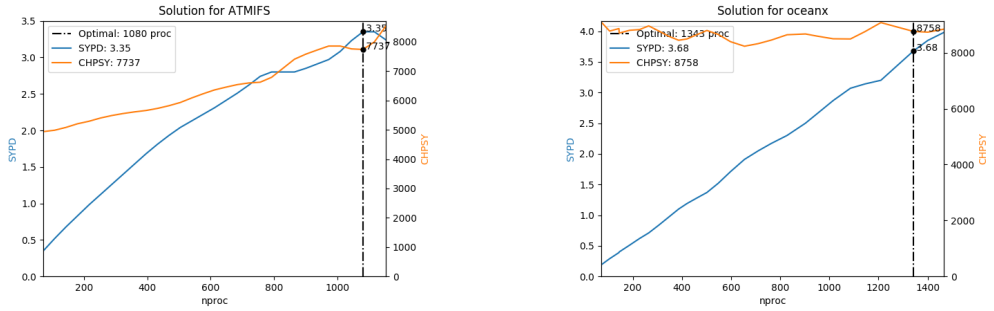
script, using 756 processes for IFS and 996 for NEMO, giving a total number of PEs of 1752. The performance of this new configuration is 2.60 SYPD and 16502 CHSY. Which means that it is 34% faster and reduced the execution cost by 6.7%. Moreover, the cost due to the coupling is reduced to 10.53%, meaning that this configuration is more balanced than the original.

Note that the workflow was also capable of balancing the default configuration (test 5). In lb-iter 3, the coupling cost is reduced to 10.94% and the performance is 2.04 SYPD and 16926 CHSY using 792 processes for NEMO and 612 for IFS. While the total number of PEs is the same, this new configuration is better than the default (5% faster and 4.5% less costly).

It is also important to note that most of the resource configurations coming from the Prediction script (tests 0-4) are much better than any of the achieved from the original configuration (test 5), even when it is balanced properly (lb-iter 3 test 5). This means that the Prediction script has found a region where IFS and NEMO can use PEs counts closer to their individual optimal scalability while still being balanced, thus proving the usefulness of the Prediction script and the FN metric.

	1	2	3	4	5	orig
IFS	1080	1008	1008	756	828	504
NEMO	1343	1086	1105	996	996	900

Table 7.1: Top 5 initial configuration from the Prediction script to be used by the load-balance workflow for a HR experiment running in the ECMWF supercomputer and a TTS_r of 0.5.



(a) (b)

Figure 7.1: Scalability and predicted best PEs allocation for IFS and NEMO in HR running on CCA machine

Figure 7.2: Performance results of each of the resource configurations tested to optimize a HR experiment in ECMWF. The metrics are the average of 2 runs of 2 months each and using a TTS_r of 0.5.

7.1.2 Standard resolution experiment in ECMWF machine

Figure 7.3a shows the scalability curve of IFS in SR running on the ECMWF machine. After 600 PEs the SYPD can not be improved much. The maximum SYPD achieved by this component is less than 20. Figure 7.3b shows the scalability results for NEMO. Until 200 PEs, the model scales linearly. After that, the execution cost when allocating more resources starts to grow and the component can not scale after 600 PEs. But with that number of PEs, the model can run at roughly 40 SYPD, doubling the speed of IFS.

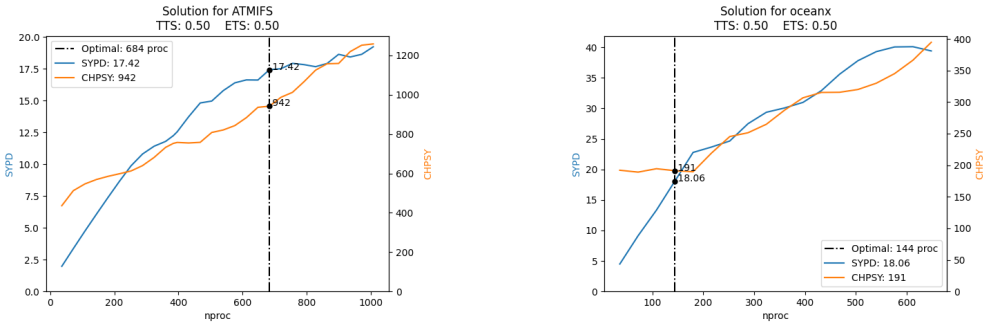
Table 7.2 shows the top5 best predicted resource configurations using a TTS_r of 0.5, without any limitation on the maximum number of resources, and using a step of one node (36 processes). The last row corresponds to the recommended set-up (orig).

Figure 7.4 The original configuration gives a SYPD of 11.22 using 432 processes (288 IFS + 144 NEMO), a CHSY of 928 and a cost due to the load-imbalance of 15.83%. The optimal solution is found by the workflow in lb-iteration 2 test 0. It uses 684 processes for IFS and 216 for NEMO, giving a total of 900 PEs for the run. The SYPD achieved by this configuration is 17.55 and the CHSY is 1230. Compared to the original

resource configuration, the best one using the Load_balance workflow is 56% faster while increasing the computational cost by 33%. Moreover, the cost due to the coupling is reduced to 11.21%, meaning that this configuration is more balanced than the original.

	1	2	3	4	5	orig
IFS	756	684	756	684	792	288
NEMO	144	144	180	180	144	144

Table 7.2: Top 5 initial configuration from the Prediction script to be used by the load-balance workflow for a HR experiment running in the ECMWF supercomputer using a TTS_r of 0.5.



(a) (b)

Figure 7.3: Scalability and predicted best PEs allocation for IFS and NEMO in SR running on ECMWF machine using a TTS_r of 0.5

Figure 7.4: Performance results of each of the resource configurations tested to optimize a SR experiment in ECMWF using a TTS_r of 0.5. The metrics are the average of 3 runs of 3 months each.

7.1.3 An ETS configuration for SR in ECMWF

In the previous section we have shown the performance improvements on a SR experiment running in ECMWF. But the difference in the total PEs used in both configurations was quite different. The original configuration used 432 processes and offered a solution which was more energy-aware, while the optimal solution found used 900 PEs and was faster but more costly.

To make fair comparison, we now take advantage of the parametrization of the Fitness metric to find a more ETS configuration by reducing the TTS_r to 0.2 (ETS_r = 0.8). Table 7.3 shows the new top 5 configurations provided by the Prediction script. Figure 7.5 shows the execution of the different tests to find the optimal. The best configuration is in lb-iter 3 test 2, using 423 PEs for IFS and 117 for NEMO. Achieving 13.94 SYPD and 939 CHSY, with a coupling cost of 8.29 and using a total of 540 PEs. Compared to the original configuration (lb-iter 0 test 5 in Figure 7.4) this new configuration is 24% (13.94/11.22) faster and only increases the execution cost by 1% (939/929). Furthermore, this new resource configuration is more balanced configuration, as the coupling cost is reduced from 15.83% to 8.29%.

	1	2	3	4	5
IFS	576	468	432	468	540
NEMO	144	108	108	144	144

Table 7.3: Top 5 initial configuration from the Prediction script to be used by the load-balance workflow using an ETS_r of 0.8 in SR running in the ECMWF supercomputer.

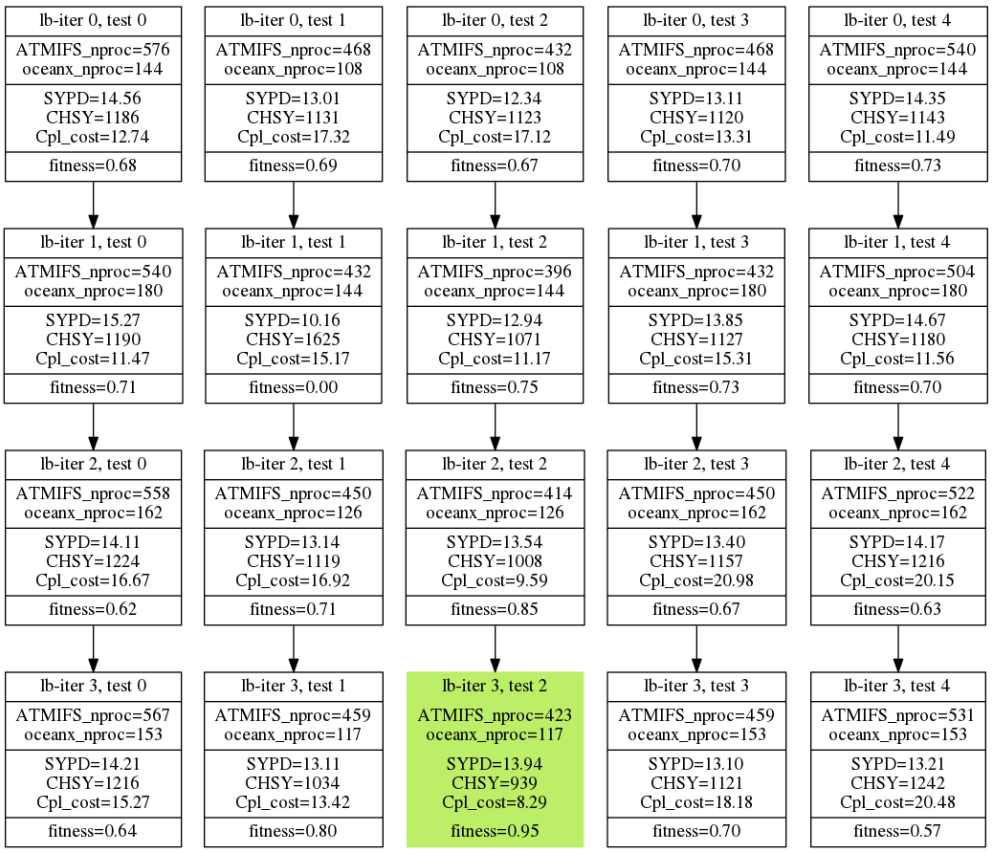


Figure 7.5: Performance results of each of the resource configurations tested to optimize a SR experiment in ECMWF using a ETS_r of 0.8. The metrics are the average of 3 runs of 3 months each.

7.2 Comparing against "optimal" solutions

The following sections shown how the auto-lb method performs against previously thought "optimal" resource configurations: A HR experiment used in the EUPC project and a SR experiment used during the CMIP6 activities. Both experiments were carefully set up so that they had exactly the same parameters and model releases as the ones used during the production runs.

7.2.1 HR EUCP

During the European Climate Prediction system (EUCP) project, a high resolution experiment involving IFS and NEMO was used to simulate a total of 400 years. The configuration used for those experiments was 912 PEs for IFS and 1392 PEs for NEMO. Figure 7.6a shows the scalability of IFS. The CHSY does not increase much up to ~500 processes. Almost achieving an ideal speedup. After 500 processes, there seems to be

	1	2	3	4	5	orig
IFS	864	912	864	768	768	912
NEMO	1389	1389	1437	1341	1389	1392

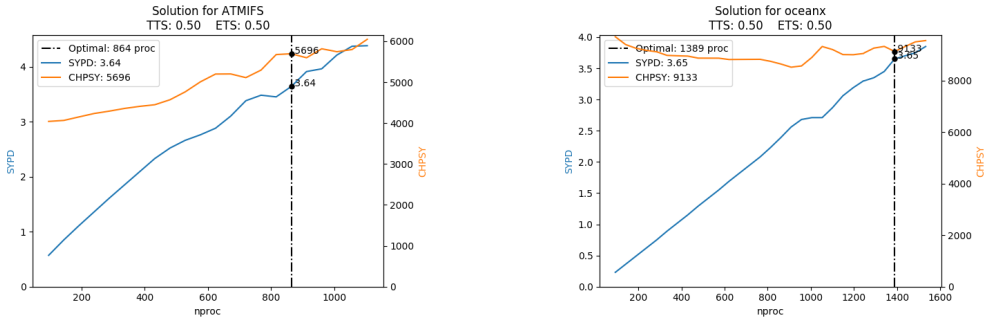
Table 7.4: Top 5 initial configuration from the Prediction script to be used by the load-balance workflow to find a better resource configuration for the EUPC HR experiment using a TTS_r of 0.5.

some number of PEs better than others as the model SYPD curve is flat around 800, 900 and over 1000 PEs. Figure 7.6b shows the scalability of NEMO. We observe a superlinear speedup as the CHSY is reduced as the number of PEs increases. The component, however, does have a sub-optimal region close to 1000 PEs. And the execution cost starts increasing at the highest number of PEs configurations.

Table 7.4 shows the default and the best 5 resource configurations found by the Prediction script. The max_nproc allowed is 2400, the TTS_r was set to 0.5, the information per timestep was provided, and there was not any nproc_restriction. The node_size was set to 48 (MareNostrum4 number of cores per node). The load-balance workflow finishes after 5 iterations. The performance metrics of each resource configuration are taken by averaging 2 runs of 2-months. The total execution time of the workflow is 50 hours (1 HourperTest * 2 TestsperConfiguration * 5 InitialConfigurations * 5 lb-iterations = 50 hours).

Figure 7.7 shows the result of the auto-lb method for this configuration. The performance of the original resource configuration was 3.54 SYPD and 16277 CHSY, with a coupling cost of 7.25%. The best solution is found in lb-iter 4 test 4 and achieves a performance of 3.48 SYPD and 15494 CHSY. This configuration is 1.7% slower than the original but reduces the execution cost by 4.9%. Moreover, note that there is also a new and better resource configuration found while trying to reduce the coupling cost for the original one, the lb-iter 3 test 5. Which uses 876 processes for ISF and 1428 for NEMO. The parallelization and the SYPD are the same as the original one but the CHSY is reduced by 363 (2.2%).

Having used this experiment to simulate the 400 years, reducing the CHSY by 4.9% is equivalent to save the executing cost of running 400*4.9% $\simeq$ 20 years (and more than 300,000 core-hours) with the same configuration.



(a) (b)

Figure 7.6: Scalability and predicted best PEs allocation for IFS and NEMO in HR for an EUCP experiment using a TTS_r of 0.5

Figure 7.7: Performance results of each of the resource configurations tested to optimize a HR EUCP experiment using a TTS_r of 0.5. The metrics are the average of 2 runs of 2 months.

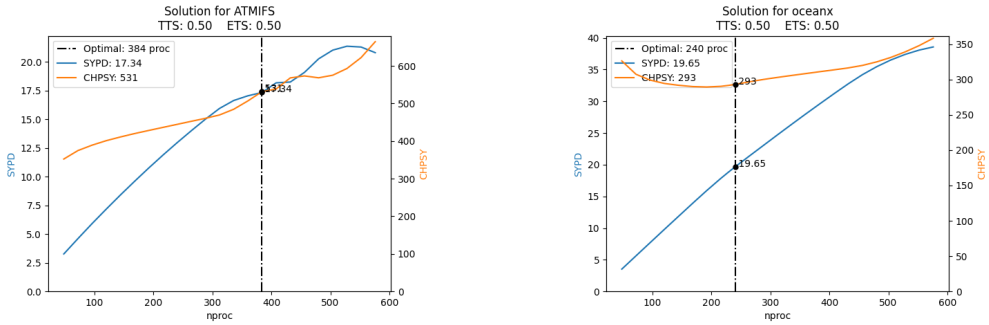
7.2.2 SR CMIP6

During the CMIP6 project, even when accounting only for experiments used for production (not taking into account the spin-up runs), more than 240000 years were simulated for multiple ESM and across different HPC platforms. At the BSC, EC-Earth3 a SR CMIP6 configuration was used to execute 14020 years in MareNostrum4. Thus, achieving the best performance was crucial for such a big project. A "optimal" resource configuration of 384 PEs for NEMO and 240 for NEMO was agreed upon. This configuration gave a total number of PEs lower than 768 so that each chunk could fit into the debug queue. Therefore, the waiting time on the queue was negligible but no more than 1 chunk can be executed at a time by each HPC user. The average performance results for one chunk with this configuration was 15.29 SYPD, 1113 CHSY and had a coupling cost of 14.81%. Figure 7.8a shows the scalability of IFS. The model scales well until 350 processes and seems to saturate at 550. Figure 7.8b shows that NEMO scales perfectly. The cost of adding more parallel resources is negligible until 600 PEs, where the speedup gains start being less pronounced than before.

After setting up the experiment and running the scalability curves for IFS and NEMO with the auto-scalabilty tool, the Prediction script was executed with the following parameters: there was not any nproc_restriction, all PEs multiple of half of the MareNostrum node size (24) were available for both components by setting the node_size pa-

	1	2	3	4	5	orig
IFS	384	360	408	408	408	384
NEMO	264	240	264	240	216	240

Table 7.5: Top 5 initial resource configurations from the Prediction script to be used by the load-balance workflow for the SR CMIP6 experiment.



(a) (b)

Figure 7.8: Scalability and predicted best resource allocation for IFS and NEMO in SR for CMIP6 experiments

rameter to 24, the max_nproc was set to 672 (maximum PEs for IFS and NEMO after subtracting the 95 used by XIOS and 1 used by RNF, $672+95+1=768$), and a TTS_r of 0.5.

The Prediction script found the optimal to be 384 PEs for IFS and 264 for NEMO. The top5 configurations are shown in Table 7.5. Each resource configuration was used to simulate 3 chunks of 6 months and the average result is taken to get the performance metrics. The result of the workflow is illustrated in Figure 7.9. Tests from 0 to 4 are resource configurations given by the prediction script and test 5 is the original one. The load-balance workflow finished after 4 iterations and a total of 24 (6x4) resource configurations have been tested. Note however that 4 of them are repeated (lb-iter 3, tests 0,3,4 and 5). The total execution time of the workflow has been 50 hours. The best result is 408 IFS - 240 NEMO, which compared to the original configuration is 4.7% faster (16.01/15.29) and 1.3% less costly (1099/1113). The coupling cost grows from 14.81% to 17.4% but it is compensated by using a number of PEs closer to the optimal for NEMO and IFS. If the resource configuration found by the auto-lb method have had been used during the CMIP6 exercise, being 4.7% faster is equivalent to reducing the simulated time by $14020/15.29 - 14020/16.01 =\sim$ 41days (if experiments were run by only one user) while a reduction of the cost by 1.3% is equivalent to the cost of simulating 182 years.

The results also show that the Prediction script has been very accurate as in lb-iter 0 tests 0,1 and 2 also offer resource configurations which are better than the original. Therefore, the only predicted configuration performing behind the original configuration is on test 4. Nevertheless, after two lb-iterations the auto-lb workflow achieved a new configuration (lb-iter 2, test 4) which also outperforms the original one. Likewise, lb-iter 1 test 5 shows that reallocating 48 processes from IFS to NEMO also provide a better configuration than the original.

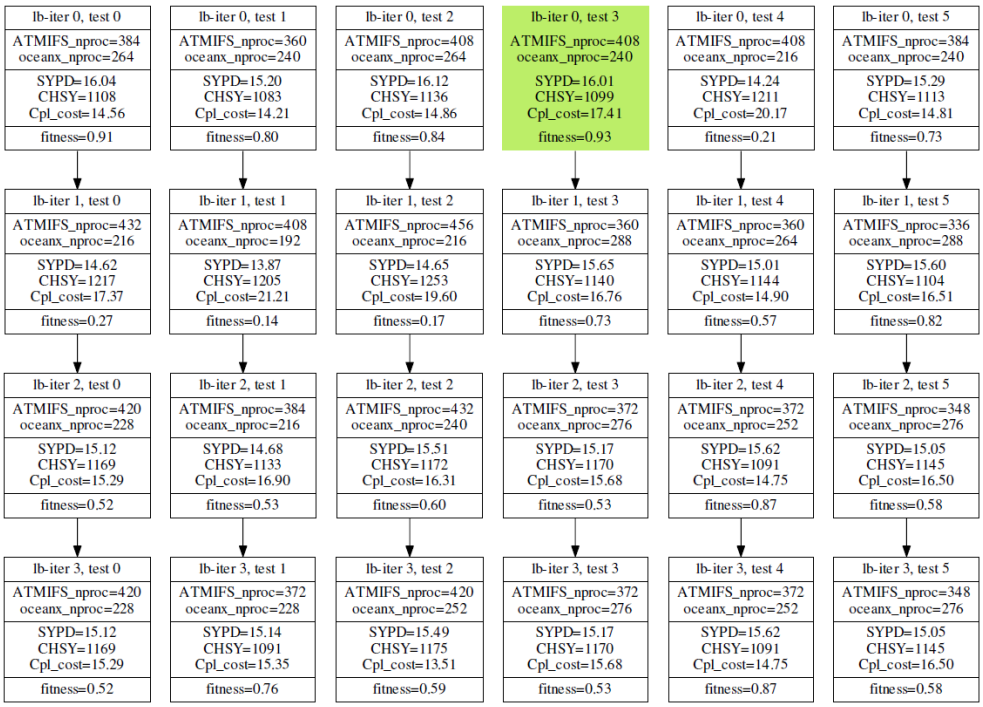


Figure 7.9: Performance results of each of the resource configurations tested to optimize a SR CMIP6 experiment. The metrics are the average of 3 runs of 6 months each.

Chapter 8

Conclusions and future work

Coupled Earth System Models (ESMs) performance is limited by the load-balance between its constituents. While some works propose to deal with the problem by adapting the applications to support malleability, operational ESMs developed and maintained by different institutions in Europe mainly try to find the best resource configurations manually. Without a proper methodology and a better set of metrics to evaluate and improve the load-imbalance, it has been shown that coupled ESMs run with suboptimal resource configurations that reduce their parallel efficiency.

This work presents a methodology to find the best number of PEs to assign to each of the components in the most used EC-Earth3 configurations. It consists of: a workflow to get the scalability properties of the coupled components, a Prediction script which can estimate what the best solutions are, and an iterative process to run the simulations on the HPC machine, gather the performance metrics and make fine-grain optimizations that reduce the coupling cost. Furthermore, the whole methodology has been integrated into the Barcelona Supercomputing Centre (BSC) official workflow manager used to run EC-Earth3, the Autosubmit workflow manager (AS), and requires the minimum user intervention possible.

To evaluate the performance of coupled ESMs we have introduced a new performance metric that allows to parameterise the energy/time tradeoff. Giving the possibility to find multiple optimal solutions depending on the specific needs a user may have: a limitation on the core-hours budged to access a HPC platform, an urgency to get the results as soon as possible, etc.

The results have shown that auto-lb can be deployed on different HPC platforms and achieve better resource configurations for multiple resolutions. In European Centre for Medium-Range Weather Forecasts (ECMWF), the method achieved to improve the speed of a HR configuration by 34% while the execution cost dropped by 6.7%. For SR experiments, we found a resource configuration 56% faster but 33% more costly. So we changed the ETS_r parameter to obtain a new allocation which is 24% faster without adding any extra execution cost.

Furthermore, it has also been shown that even resource allocations that were thought to be optimal and used for big European and Worldwide projects like Coupled Model Intercomparison Project Phase 6 (CMIP6) could be still improved, thus potentially saving 4.7% of of simulated time while still reducing the execution cost by 1.3% for SR configurations that took months to simulate and consumed over 15M core-hours in MareNostrum4.

Given that the method does not require to modify the sources of the ESMs, it can be easily ported to optimize the resource configurations for other ESMs apart from EC-Earth3. AS is used by other models to execute production and operational experiments

in the Earth Science department and will be the workflow manager adopted by the next EC-Earth version (EC-Earth4). Thus, allowing to give access to the auto-lb method easily for a broad community.

In the future, ESMs similar to EC-Earth will include more components that will be used in big projects like CMIP7. The auto-lb functionalities will be extended and be ready for configurations using over 4 components coupled in the same simulation. In this context, we strongly believe that traditional methods would not be able to achieve even suboptimal resource configurations.

Acknowledgments

This work was developed and supported by the Computational Earth Science group of the Barcelona Supercomputing Centre. I'd like to thank Autosubmit team members Miguel Castrillo and Daniel Beltran for their valuable and useful suggestions during the integration of my work to the workflow manager.

I would also like to thank Eric Ferrer for helping me with the deployment on ECMWF and to Aude Carreric for assisting me with all troubleshooting while configuring the simulations.

References

[1] Syukuro Manabe and Kirk Bryan. "Climate Calculations with a Combined Ocean-Atmosphere Model". In: *Journal of The Atmospheric Sciences - J ATMOS SCI* 26 (June 1969), pp. 786–789. DOI: 10.1175/1520-0469(1969)026<0786:CCWACO>2.0.CO;2.

[2] V Eyring et al. "Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization". In: *Geoscientific Model Development* 9 (5 2016), pp. 1937–1958. DOI: 10.5194/gmd-9-1937-2016. URL: https://gmd.copernicus.org/articles/9/1937/2016/.

[3] Gerald A Meehl et al. "The Coupled Model Intercomparison Project (CMIP)". In: *Bulletin of the American Meteorological Society* 81 (2 2000), pp. 313–318. ISSN: 00030007, 15200477. URL: http://www.jstor.org/stable/26215108.

[4] Mario Acosta, Sergi Palomas, and Stella Paronuzzi. "IS-ENES3 D4.3 - CPMIP performance metrics and community advice". In: (Dec. 2021). DOI: 10.5281/ZENODO.6394049. URL: https://doi.org/10.5281/zenodo.6394049#.YqMDPFZSAoF.mendeley.

[5] Jean-Claude André et al. "High-Performance Computing for Climate Modeling". In: *Bulletin of the American Meteorological Society* 95 (June 2014), ES97–ES100. DOI: 10.1175/BAMS-D-13-00098.1.

[6] F Cappello, W Gentzsch, and M Valero. "HPCS 2013 Panel: The era of exascale sciences: Challenges, needs and requirements". In: June 2013, p. 1.

[7] N Attig, P Gibbon, and Th. Lippert. "Trends in supercomputing: The European path to exascale". In: *Computer Physics Communications* 182 (9 2011). Computer Physics Communications Special Edition for Conference on Computational Physics Trondheim, Norway, June 23-26, 2010, pp. 2041–2046. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2010.11.011. URL: https://www.sciencedirect.com/science/article/pii/S0010465510004571.

[8] Daniel A Reed and Jack Dongarra. "Exascale Computing and Big Data". In: *Commun. ACM* 58 (7 June 2015), pp. 56–68. ISSN: 0001-0782. DOI: 10.1145/2699414. URL: https://doi.org/10.1145/2699414.

[9] Michael F Wehner et al. "Hardware/software co-design of global cloud system resolving models". In: *Journal of Advances in Modeling Earth Systems* 3 (4 2011). DOI: https://doi.org/10.1029/2011MS000073. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2011MS000073.

[10] V Balaji. "Climate Computing: The State of Play". In: *Computing in Science & Engineering* 17 (6 2015), pp. 9–13. DOI: 10.1109/MCSE.2015.109.

[11] S. Valcke et al. "Coupling technologies for Earth System Modelling". In: *Geoscientific Model Development* 5 (6 Dec. 2012), pp. 1589–1596. ISSN: 1991-9603. DOI: 10.5194/gmd-5-1589-2012. URL: https://gmd.copernicus.org/articles/5/1589/2012/.

[12] Carlos Roberto Mechoso, Soon-Il An, and Sophie Valcke. *Atmosphere-ocean Modeling: Coupling and Couplers*. World Scientific, 2021. Chap. 8.

[13] S Valcke. "The OASIS3 coupler: a European climate modelling community software". In: *Geoscientific Model Development* 6 (2 2013), pp. 373–388. DOI: 10.5194/gmd-6-373-2013. URL: https://gmd.copernicus.org/articles/6/373/2013/.

[14] Jay Larson, Robert Jacob, and Everest Ong. "The Model Coupling Toolkit: A New Fortran90 Toolkit for Building Multiphysics Parallel Coupled Models". In: *The International Journal of High Performance Computing Applications* 19 (3 2005), pp. 277–292. DOI: 10.1177/1094342005056115. URL: https://doi.org/10.1177/1094342005056115.

[15] L. Liu et al. "C-Coupler1: a Chinese community coupler for Earth system modeling". In: *Geoscientific Model Development* 7 (5 Oct. 2014), pp. 2281–2302. ISSN: 1991-9603. DOI: 10.5194/gmd-7-2281-2014. URL: https://gmd.copernicus.org/articles/7/2281/2014/.

[16] Moritz Hanke et al. "YAC 1.2.0: new aspects for coupling software in Earth system modelling". In: *Geoscientific Model Development* 9 (8 Aug. 2016), pp. 2755–2769. ISSN: 1991-9603. DOI: 10.5194/gmd-9-2755-2016. URL: https://gmd.copernicus.org/articles/9/2755/2016/.

[17] R Essery. "A factorial snowpack model (FSM 1.0)". In: *Geoscientific Model Development* 8 (12 2015), pp. 3867–3876. DOI: 10.5194/gmd-8-3867-2015. URL: https://gmd.copernicus.org/articles/8/3867/2015/.

[18] Nancy Collins et al. "Design and implementation of components in the Earth System Modeling Framework". In: *The International Journal of High Performance Computing Applications* 19 (3 2005), pp. 341–350.

[19] Tony Craig. "CPL7 user's guide". In: *Updated for CESM version* 1 (6 2014).

[20] R Döscher et al. "The EC-Earth3 Earth system model for the Coupled Model Intercomparison Project 6". In: *Geoscientific Model Development* 15 (7 2022), pp. 2973–3020. DOI: 10.5194/gmd-15-2973-2022. URL: https://gmd.copernicus.org/articles/15/2973/2022/.

[21] Dror G Feitelson and Larry Rudolph. "Parallel job scheduling: Issues and approaches". In: 1995, pp. 1–18.

[22] Sathish Vadhiyar and Jack Dongarra. "SRS - A Framework for Developing Malleable and Migratable Parallel Applications for Distributed Systems". In: *Parallel Processing Letters* 13 (June 2003). DOI: 10.1142/S0129626403001288.

[23] Kaoutar Maghraoui, Boleslaw Szymanski, and Carlos Varela. "An Architecture for Reconfigurable Iterative MPI Applications in Dynamic Environments". In: vol. 3911. June 2005, pp. 258–271. ISBN: 978-3-540-34141-3. DOI: 10.1007/11752578_32.

[24] Kaoutar El Maghraoui et al. "Dynamic Malleability in Iterative MPI Applications". In: 2007, pp. 591–598. DOI: 10.1109/CCGRID.2007.45.

[25] Daihee Kim, J Walter Larson, and Kenneth Chiu. "Toward Malleable Model Coupling". In: *Procedia Computer Science* 4 (2011). Proceedings of the International Conference on Computational Science, ICCS 2011, pp. 312–321. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2011.04.033. URL: https://www.sciencedirect.com/science/article/pii/S1877050911000913.

[26] Daihee Kim, J Walter Larson, and Kenneth Chiu. "Malleable Model Coupling with Prediction". In: 2012, pp. 360–367. DOI: 10.1109/CCGrid.2012.20.

[27] Daihee Kim, J Walter Larson, and Kenneth Chiu. "Dynamic Load Balancing for Malleable Model Coupling". In: 2012, pp. 150–157. DOI: 10.1109/ISPA.2012.28.

[28] Daihee Kim, J Walter Larson, and Kenneth Chiu. "Automatic Performance Prediction for Load-Balancing Coupled Models". In: 2013, pp. 410–417. DOI: 10.1109/CCGrid.2013.72.

[29] Yuri Alexeev et al. "The Heuristic Static Load-Balancing Algorithm Applied to the Community Earth System Model". In: 2014, pp. 1581–1590. DOI: 10.1109/IPDPSW.2014.177.

[30] Ding Nan et al. "CESMTuner: An Auto-tuning Framework for the Community Earth System Model". In: 2014, pp. 282–289. DOI: 10.1109/HPCC.2014.51.

[31] Hongliang Li et al. "CPSA: A Coordinated Process Scheduling Algorithm for Coupled Earth System Model". In: 2020, pp. 1–9. DOI: 10.1109/ICCCN49398.2020.9209733.

[32] Marta Garcia, Julita Corbalan, and Jesus Labarta. "LeWI: A Runtime Balancing Algorithm for Nested Parallelism". In: 2009, pp. 526–533. DOI: 10.1109/ICPP.2009.56.

[33] Garcia Marta et al. *A Dynamic Load Balancing Approach with SMPSuperscalar and MPI*. Ed. by David, Weiss Jan-Philipp Keller Rainer, and Kramer. 2012. DOI: 10.1007/978-3-642-30397-5_2. URL: https://doi.org/10.1007/978-3-642-30397-5_2.

[34] Andreas Will et al. "The COSMO-CLM 4.8 regional climate model coupled to regional ocean, land surface and global earth system models using OASIS3-MCT: description and performance". In: *Geoscientific Model Development* 10 (4 Apr. 2017), pp. 1549–1586. ISSN: 1991-9603. DOI: 10.5194/gmd-10-1549-2017. URL: https://gmd.copernicus.org/articles/10/1549/2017/.

[35] M.C. Acosta et al. *Performance analysis of EC-Earth 3.2: Coupling*. BSC-CNS, 2016. URL: https://earth.bsc.es/wiki/lib/exe/fetch.php?media=library:external:technical_memoranda:bsc-ces-2016-006-coupling_ec-earth.pdf.

[36] V Balaji et al. "CPMIP: measurements of real computational performance of Earth system models in CMIP6". In: *Geoscientific Model Development* 10 (1 2017), pp. 19–34. DOI: 10.5194/gmd-10-19-2017. URL: https://gmd.copernicus.org/articles/10/19/2017/.

[37] R Gonzalez and M Horowitz. "Energy dissipation in general purpose microprocessors". In: *IEEE Journal of Solid-State Circuits* 31 (9 1996), pp. 1277–1284. DOI: 10.1109/4.535411.

[38] Sarah Abdulsalam et al. "Using the Greenup, Powerup, and Speedup metrics to evaluate software energy efficiency". In: 2015, pp. 1–8. DOI: 10.1109/IGCC.2015.7393699.

[39] Xavier Yepes-Arbós et al. *Scalability and performance analysis of EC-EARTH 3.2.0 using a new metrics approach (part I)*. BSC-CNS, 2016.

[40] Domingo Manubens-Gil et al. "Seamless management of ensemble climate prediction experiments on HPC platforms". In: 2016, pp. 895–900. DOI: 10.1109/HPCSim.2016.7568429.